Optimal Sampling for Hemicubes

Nelson Max
University of California, Davis, and
Lawrence Livermore National Laboratory

Abstract

The hemicube estimates of form factors are based on a finite set of sample directions. We obtain several optimal arrangements of sample directions, which minimize the variance of these estimates. They are based on changing the size or shape of the pixels or the shape of the hemicube, or using non-uniform pixel grids. The best reduces the variance by 43%.

The variance calculation is based on the assumption that the errors in the estimate are caused by the projections of single polygon edges, and that the positions and orientations of these edges are random. This replaces the infinite dimensional space of possible environments by the two dimensional space of great circles on the unit sphere, making the numerical variance minimization possible.

Keywords: hemicube, radiosity, form factor, sampling, variance, optimization

Introduction

Radiosity algorithms for global illumination, either “gathering” [1,2] or “shooting” [3] versions, depend on the calculation of form factors. It is possible to calculate the form factors analytically [1,4,5,6,7], but this is difficult when occlusion is involved, so sampling methods are usually preferred. The necessary visibility information can be obtained by ray tracing in the sampled directions. However, area coherence makes it more efficient to project and scan-convert the scene onto a number of planes, for example, the faces of a hemicube[2]. The hemicube faces have traditionally been divided into equal square pixels, but more general subdivisions are practical, and can reduce the variance of the form factor estimates.

Sillion and Puech [8] used a single horizontal plane instead of a hemicube, and suggested distributing the samples more densely in directions near the surface normal, in order “to obtain regions with equal contributions to the form factor”. Recker *et al*. [9] also try to do this in a progressive radiosity context by using a higher resolution central region on the single plane, and later shooting from the missed vertical sides of a short, wide hemicube.

The main innovation here is a quantification of the effect of the distribution of sample directions, which produces an optimization condition somewhat different than the “equal contribution” one quoted above, due to area coherence effects. Optimization results in several “recipes” for distributing the sample directions on the hemicube. The recipes are resolution-independent. Given the number K of hemicube pixel samples desired, the recipe produces an arrangement of close to, but no more than, K samples. The recipes were tested on random input, and shown to be superior to one optimized for “equal contributions to the form factor”.

The optimization uses no knowledge about the specific input geometry, and instead attempts to reduce the expected variance in the form factors for random input scenes. Thus the recipes are not adaptive. They are fixed, independent of the input, and amenable to hardware speedups.

Adaptive sampling is appropriate for ray tracing, since rays can be independently positioned at no extra cost. However a hemicube z-buffer algorithm taking advantage of area coherence during scan conversion requires a fixed sampling pattern. The patterns proposed here can all use efficient hardware or software scan conversion. The best uniform grid approach, compatible with current hardware, is predicted to reduce the variance by 31%, compared with the standard hemicube. A non-uniform grid based on cubic polynomials is predicted to reduce the variance by 43%, but requires either software scan conversion or revised hardware microcode to perform an extra table lookup and multiplication per pixel.

Since the optimization assumes random inputs, we must understand the probability distribution on the space of scene geometries. With no limit on scene complexity, this space is infinite dimensional, so simplifications are required. The key simplifying idea here is that when surfaces are broken up into polygonal patches, and hemicubes are rotated randomly about the surface normals to reduce form factor aliasing [10], the relationship of the patch edges to the hemicube becomes random. Random edges in 3D project to random great circles on the unit sphere *U* of possible sample directions. The principal errors in the form factor estimates can be directly related to the positions of these great circles. This reduces the distribution of scene geometries to the much simpler two-parameter distribution of random great circles on *U,* and makes the mathematical analysis practical.

Form Factor Estimates

We briefly define form factors and describe hemicubes below, and then proceed to analyze the errors inherent in hemicube sampling, and how to minimize them. (Readers who require motivation to traverse the mathematics should skip to the results section first.)

To obtain the form factor $F_{dA_i - A_j}$ between a finite area A_j and a differential area dA_i, one needs to calculate an integral [4],

$$F_{dA_i - A_j} = \int_{A_j} V(dA_i, dA_j) \frac{\cos\theta_i \cos\theta_j}{\pi r^2} dA_j , \tag{1}$$

where $V(dA_i,dA_j)$ is 1 if the differential area dA_j is visible from dA_i, and 0 otherwise, r is the length of the ray R from dA_i to dA_j, θ_i is the angle between the ray R and the normal to dA_i, and θ_j is the angle between R and the normal to dA_j. This integral is equivalent to

$$F_{dA_i - A_j} = \frac{1}{\pi}\int_H V'(\omega, A_j) \cos\alpha(\omega)\, d\omega \tag{2}$$

where H is the hemisphere of unit direction vectors ω above dA_i, $d\omega$ is the differential solid angle on this hemisphere, and $\alpha(\omega)$ is the angle between the direction ω and the normal to dA_i , the same as θ_i in (1), and $V'(\omega,A_j)$ is 1 if the surface visible from dA_i in the direction ω is A_j, and is 0 otherwise. The integral (2) can be calculated analytically, using exact visibility algorithms [4,5]. However it is usually estimated as a Riemann sum, by dividing the hemisphere H into a number of disjoint regions R_k of solid angle $\Delta\omega_k$:

$$F'_{dA_i - A_j} = \frac{1}{\pi}\sum_k V'(\omega_k, A_j) \cos\alpha(\omega_k)\, \Delta\omega_k \tag{3}$$

where ω_k is a sample direction inside R_k, usually at its center. A slightly more accurate estimate is the weighted sum

$$F''_{dA_i - A_j} = \sum_k V'(\omega_k, A_j)\, W_k \tag{4}$$

where

$$W_k = \frac{1}{\pi}\int_{R_k} \cos\alpha(\omega)\, d\omega\,.$$

The estimate (4) will be correct if the set where $V'(\omega, A_j) = 1$ aligns exactly with a collection of the regions R_k, while (3) may still be in error.

In the hemicube algorithm of Cohen and Greenberg [2], the regions R_k are the projections onto the unit hemisphere H of square pixels (also called R_k below) on the faces of a half of a cube C surrounding H, and the weights W_k correspond to the “Δ form factors”. By scan converting the projections of all surfaces A_j onto the faces of C, using a Z-buffer or another standard visibility algorithm, an “item buffer” B can be prepared, such that $B(k) = j$ if and only if $V'(\omega_k, A_j) = 1$. The $F''_{dA_i - A_j}$ can then easily be obtained from the item buffer as

$$F''_{dA_i - A_j} = \sum_{B(k) = j} W_k \tag{5}$$

Ray tracing algorithms [6] for calculating form factors can also be put into this framework by taking the ω_k to be the sample ray directions, and R_k to be the subset of the unit sphere containing those directions closer to sample ω_k than to any other sample direction.

Obviously, the accuracy of the estimate (4) depends on the number of samples ω_k, and their arrangement on the hemisphere H. For a given number K of samples, our goal is to find the best arrangement. Special attention will be paid to arrangements compatible with hardware rendering engines, or with software scan conversion algorithms that take advantage of object coherence.

Error Statistics

The difference between the integral (2) and its estimate as the sum (4) is the error

$$D = F_{dA_i - A_j} - F''_{dA_i - A_j} = \frac{1}{\pi}\int_H \{V'(\omega, A_j) - V'(\omega_k(\omega), A_j)\} \cos\alpha(\omega)\, d\omega \tag{6}$$

where $\omega_k(\omega)$ is the sample ray corresponding to the region R_k containing the direction ω.

Any sampling method of estimating an integral like (2) is subject to error. The estimate (4) comes from assuming that the polygon visible at the sample point ω_k is actually visible inside the whole region R_k. Thus it puts a jagged staircase edge following the hemicube pixel boundaries in place of the actual boundary edges of the visible regions, causing aliasing.

Note that at this stage in a radiosity algorithm, our goal is not to reconstruct an image of the scene from the point of view of the hemicube center. All we need is an accurate estimate of the form factor $F_{dA_i - A_j}$. If a protrusion in a visible region making the estimate too large is compensated by a nearby intrusion making it a compensating amount too small, the estimate will still be correct. Since this cancellation does happen on average, our estimate (4) is unbiased: if we repeat the calculation many times with randomly rotated hemicubes, the average will approach the correct integral (2).

For hemicube rotation by a random angle θ, the expected value of a quantity *h* depending on θ is

$$E_\theta (h) = \frac{1}{2\pi} \int_0^{2\pi} h(\theta)\, d\theta .$$

The fact that $F''_{dA_i - A_j}$ is unbiased means $E_\theta (D) = F_{dA_i - A_j} - E_\theta (F''_{dA_i - A_j}) = 0$. In spite of this, any one calculation will likely be in error. A measure of the range of this error is the variance of $F''_{dA_i - A_j}$, which is the expectation of its squared deviation from its expected value, or $E_\theta \left((F''_{dA_i - A_j} - E_\theta (F''_{dA_i - A_j}))^2 \right) = E_\theta \left(D^2 \right)$. Our goal is to minimize this variance, and thus reduce as much as possible the errors caused by aliasing.

Many variance reduction techniques have been suggested for ray tracing, such as adaptive supersampling (Whitted [11]) and stratified sampling (Lee *et al*. [12]). However our goal is a non-adaptive method suitable for hardware scan conversion into a z-buffer. This offers a tremendous speedup from the area coherence in the scan conversion. In addition, a single scan through the item buffer can produce an estimate (5) for all possible A_j.

Suppose the regions A_j are totally and randomly spread out, as in a Jackson Pollack spatter painting, with paint drops smaller than the hemicube grid spacing, and A_j corresponding to the union of all spots of color j. Then the only way to estimate the integral (2) is by Monte Carlo sampling. The samples are uncorrelated, and the all have the same probability p_j of hitting a spot of color j. The visibility function $V'(\omega,A_j)$ also has the same variance $v_j = p_j - p_j^2$ at every sample ω, and a simple analysis shows that if n samples are taken, the variance of their mean visibility is $v_j / n = O(n^{-1})$.

In our case, however, the regions being sampled are polygons, which means that nearby samples are correlated. I will show below that a regular pattern of samples can take advantage of this fact, and produce a variance of order $O(n^{-3/2})$, which is a significant improvement over random sampling.

Another method for numerically estimating the integral (2) is Gauss integration. (See Burnet [13], Zatz [14].) The integrand is evaluated at a number of specially placed Gauss points along A_j, and the estimate is a weighted sum of the results. The Gauss points and their weights are chosen to integrate exactly polynomials of up to a certain maximum degree. The method thus works well on smooth functions that are approximated well by polynomials. However the visibility function $V(dA_i, A_j)$ changes discontinuously at occlusion edges that hide parts of A_j from dA_i, so Gauss integration cannot be expected to be accurate. For similar reasons Simpson’s rule is not useful.

Calculation of variance

To compute $E_\theta(D^2)$ we start by rewriting D as a sum $D = \sum_k D_k$ where

$$D_k = \frac{1}{\pi}\int_{R_k} \{ V'(\omega, A_j) - V'(\omega_k, A_j) \} \cos\alpha(\omega)\, d\omega . \tag{7}$$

Then

$$E_\theta (D^2) = \sum_k \sum_l D_{kl} \tag{8}$$

where the “error correlation” $D_{kl} = E_\theta (D_k D_l)$.

Instead of being fixed, suppose that a polygonal input scene is chosen from a distribution of possible inputs. Suppose that A_j is selected randomly from the patches in this scene, that dA_i is chosen randomly from among the surface points at which form factors are required, and, as above, that the hemicube is rotated randomly about the normal to dA_i. Together these choices define a probability distribution dg on the space G of possible geometry affecting the form factor $F_{dA_i - A_j}$ and its estimate $F''_{dA_i - A_j}$.

If h is a function of geometry g, we will use $E(h)$ without the subscript θ to denote the expected value

$$E(h) = \int_G h(g)\, dg\,.$$

Now $E(D^2)$ is the expected variance for an arbitrary form factor for a geometry in G, and this is what we will try to minimize.

Consider the visible projection $P(A_j)$ of A_j on the hemisphere H, *i.e.*, $P(A_j) = \{\omega \mid V'(\omega, A_j) = 1\}$. In the case of polygonal input, $P(A_j)$ is bounded by projections of straight line polygon edges, or projections of straight lines in which two polygons intersect. Note that D_k in (7) is zero unless R_k is crossed by one of these boundary edges. For random geometries, D_{kl} becomes smaller and smaller as the regions R_k and R_l grow farther apart, because it becomes less and less likely that both regions will be crossed by a boundary edge. For fixed R_k and fixed dr, the number of region samples ω_l whose distance to ω_k is between r and $r + dr$ increases linearly with r, so many small terms D_{kl} for distant pairs of regions might still contribute substantially to the sum (8) for $E(D^2)$. However, detailed analysis like that in the following sections shows that as r increases, the positive and negative contributions to the integral over G for D_{kl}, from edges of varying position and orientation, cancel so as to make D_{kl} decrease rapidly enough that only neighboring regions R_k and R_l contribute significantly to $E(D^2)$. Therefore we are only interested in the D_{kl} for neighboring regions.

In addition, R_k and R_l should be small compared to the size of $P(A_j)$, or the estimate (4) will have a large relative error. The calculations below are for the limiting case where the hemicube subdivision is fine compared to the projected size of the polygons. Under these conditions it is reasonable to make our “first basic assumption”, that at most a single edge L of $P(A_j)$ crosses R_k or R_l, since other situations will contribute insignificantly to the variance.

If the distribution dg of geometries is random, the 3-D edge corresponding to L will be a random line in space, so L will be a random great circle on the hemisphere H. Thus our “second basic assumption” is to replace the integration over G in $E(D^2)$ by integration over the space S of great circles. This space can be parametrized by the unit normal to the plane of the great circle, the one which points to the side containing the polygon A_j. The probability measure of a collection of great circles is then proportional to the area on the unit sphere occupied by the corresponding normals.

We have seen above that only terms D_{kl} for neighboring pairs of regions crossed by the same edge of $P(A_j)$ contribute significantly to $E(D^2)$. When the hemicube grid is fine, the number of these pairs is proportional to the total boundary edge length of $P(A_j)$. Thus the effect of this second assumption is to multiply the variance $E(D^2)$ by the ratio of the length of a complete great circle to the sum of the lengths of the great circle arcs bounding $P(A_j)$. A distribution of samples which is optimal for random great circles will also be optimal for random polygons. This makes it possible to account for the randomness in the position and orientation of the polygons, without worrying about the distribution of their sizes. We will quote our variance values for the single great circle case.

The assumption of random orientation may not be appropriate for architectural simulations, where the edges are preferentially oriented along three perpendicular axes. Randomness can be partially restored by rotating the hemicube, randomly around the surface normal, as suggested by Wallace *et al.* [10]. This will randomize the edges which are not parallel to the surface normal, but only the partially randomize edges which are parallel to it. In the "Future work" section, I suggest how to modify the analysis for this case.

Note that if $V'(\omega, A_j)$ is replaced by a function which is linear across R_k, the deviations from the value at a region center are symmetric, and cancel out in the integral (7) for $D_{k,}$ so the main source of variance still comes from projected edges. This makes our analysis applicable to the linear finite elements in Max and Allison [15], the piecewise linear cases of the more general finite element formulations in Troutman and Max [16], Zatz [14], and Gortler *et al.* [17], and to the final light-gathering pass in Cohen and Greenberg [2]. In the work of Chen *et al.* [18], the final gathering is done once per output pixel, so the variance results in noise in the final rendering, which can also be reduced by optimal sampling.

In finite element applications, the point collocation method corresponds to the finite-area-to-differential-area form factors $F_{dA_i - A_j}$, which can be found using hemicubes, while the more popular Galerkin method corresponds to the finite-area-to-finite-area form factors$F_{A_i - A_j}$, which involve integrals over four real variables. Troutman and Max [16] found that the point collocation method converged faster than the Galerkin method, even when the hemicube sampling was done in software. The 4D integrals for the Galerkin method are usually done by Gauss integration, but as discussed above, this calculation may be inaccurate due to the effect of discontinuities. A hybrid integration method could place hemicubes at the Gauss points dA_i of A_i, and use 2D Gauss integration only over A_i. In general occlusion situations, $F_{dA_i - A_j}$ is a C^2 function of the position of dA_i, with lower order continuity coming only from degenerate situations when edges are parallel or touch (Paul Heckbert, personal communications), so 2D Gauss integration should perform better over A_i. As shown in Max and Allison [15], the color channels in a hardware pipeline can be used to help integrate all the piecewise linear basis functions in one pass over a hemicube at dA_i.

Variance from a central rectangular pixel

We will calculate the D_{kl} for regions on the plane T of the top hemicube face, tangent to H at the north pole O, because polygon edges project to straight lines on T, rather than to great circles. We will replace the space S of great circles by the simpler space Q of all oriented lines L, randomly positioned in the plane T. In the following section, we will show how to compensate for the resulting distortion in the distribution of random lines. For now, we will consider only regions close to the point O of tangency between T and H, where this distortion is minimal.

We will start by analyzing the variance D_{kk} contributed by a single rectangular pixel R_k, whose center is O. The rectangle R_k, shown in figure 1, has width $2a$ and height $2b$, and is bounded by the edges $x = a$, $x = - a$, $y = b$, and $y = - b$. The space Q of lines will be parametrized by the length l of the segment OG from O perpendicular to L, and the angle θ between this segment and the X axis. The line $L(\theta, l)$ is oriented so that $V'(\omega, A_j)$ is 1 on its right side, and 0 on its left. The parameter l is positive if O is to the left of the directed line L, the case shown in figure 1, and negative otherwise. For now, both θ and l have uniform distributions. Rewriting (8) for $l= k$,

$$D_{kk} = E\left(\frac{1}{\pi}\int_{R_k} \{ V'(\omega, A_j) - V'(\omega_k, A_j) \} \cos\alpha(\omega)\, d\omega \right)^2 \qquad (9)$$

At the north pole, cos $\alpha = 1$, and the solid angle $d\omega$ is equivalent to the area $dx\ dy$. Thus D_{kk} is well approximated by

$$E_{kk} = \frac{1}{\pi^2} \int_0^{2\pi} d\theta \int_{-\infty}^{\infty} dl\, (D(\theta, l))^2 \tag{10}$$

where

$$D(\theta, l) = \int_{-a}^{a} dx \int_{-b}^{b} dy \{ V''(x, y) - V''(0, 0) \} \tag{11}$$

and $V''(x, y)$ is 1 on the right side of $L(\theta, l)$, and 0 on the left.

The space S of great circles is finite, but the range of l in (10) is infinite, so the space Q is infinite, and the area $d\theta\ dl$ cannot be normalized into a probability distribution. We will fix this later when we correct for the area shrinkage in the mapping from Q to S, but for now, note that $D(\theta, l)$ is 0 whenever l is large enough so that the line does not cross the rectangle R_k.

Note that adding π to θ and reversing the sign of l puts $L(\theta, l)$ back in the same place, with its orientation reversed. Since line L is oriented to make the interior of $P(A_j)$ lie to its right, this replaces V'' by 1 - V'' and changes the sign of $D(\theta, l)$, but does not change its square. Similarly, reflecting figure 1 in the X or Y axis leaves $(D(\theta, l))^2$ unchanged. Thus

$$E_{kk} = \frac{8}{\pi^2} \int_0^{\pi/2} d\theta \int_0^{\infty} dl\, (D(\theta, l))^2 \tag{12}$$

so it is sufficient to compute $D(\theta, l)$ for θ in the first quadrant, and l positive, so that the line L is oriented as in figure 1, and $V''(0, 0) = 0$. The equation of the line $L(\theta, l)$ is

$$e(x, y) = x\cos\theta + y\sin\theta - l = 0 \tag{13}$$

and with the orientation in figure 1, $V''(x, y) = 1$ if and only if $e(x, y) \geq 0$. The maximum value of l giving a non-zero $D(\theta, l)$, for θ in the first quadrant, is at

$$l_1(\theta) = a\cos\theta + b\sin\theta \tag{14}$$

where the corner $B = (a, b)$ satisfies (13). For $l > l_1(\theta)$, the line L misses R_k.

There are two other special cases for the intersection topology of L with R_k. When the line L passes through the corner $A = (- a, b)$, l is at

$$l_2(\theta) = -a\cos\theta + b\sin\theta \tag{15}$$

and when L passes through $D= (a, - b)$, l is at

$$l_3(\theta) = a\cos\theta - b\sin\theta \tag{16}$$

Note that (15) gives a positive $l_2(\theta)$ only when $\theta > \theta_3 = \tan^{-1}(a/b)$, and (16) gives a positive $l_3(\theta)$ only when $\theta < \theta_3$.

The three "general position" cases for the intersection of L with R_k are shown in figures 1, 2, and 3, and their (θ, l) ranges are shown in figure 4. The situation in figure 1, or in the limiting special cases for which the same area formulas hold, occurs when $\max(l_2, l_3) \leq l \leq l_1$. In this case, $D(\theta, l) = D_1(\theta, l)$, the area of the triangle EBF where $V''(x, y) = 1$. In figure 1, $BS = l_1 - l$, $EB = BS / \cos\theta$, and $FB = BS / \sin\theta$, so

$$D_1(\theta, l) = \frac{1}{2}EB \cdot FB = \frac{(l_1 - l)^2}{2\cos\theta\sin\theta}.$$

Case 2, shown in figure 2, occurs when $\theta \geq \theta_3$ and $0 \leq l \leq l_2(\theta)$. In this case, $D(\theta, l) = D_2(\theta, l)$, the area of the trapezoid $EABF$, with base AB and average height UT. The distance OT is $l / \sin\theta$ so $UT = OU - OT = b - l / \sin\theta$, and

$$D_2(\theta, l) \;=\; AB \cdot UT \;=\; \frac{2a(b\sin\theta - l)}{\sin\theta}.$$

Case 3, shown in figure 3, occurs when $\theta \le \theta_3$ and $0 \le l \le l_3(\theta)$. Case 2 can be transformed into case 3 by reflection in the 45° line $x = y$, which swaps a and b, and also swaps $\cos\theta$ and $\sin\theta$, so

$$D_3(\theta, l) \;=\; \frac{2b(a\cos\theta - l)}{\cos\theta}.$$

I used Mathematica™ to integrate $(D(\theta, l))^2$, first in l and then in θ, using the different formulas above for the different regions shown in figure 4. For fixed $\theta \le \theta_3$,

$$\int_0^{\infty} (D(\theta, l))^2 dl \;=\; \int_0^{l_3(\theta)} (D_3(\theta, l))^2 dl + \int_{l_3(\theta)}^{l_1(\theta)} (D_1(\theta, l))^2 dl$$

$$= \frac{4}{15} b^5 \frac{\sin^3\theta}{\cos^2\theta} + \frac{4}{3} a^3 b^2 \cos\theta \qquad (17)$$

and for $\theta \ge \theta_3$,

$$\int_0^{\infty} (D(\theta, l))^2 dl \;=\; \int_0^{l_2(\theta)} (D_2(\theta, l))^2 dl + \int_{l_2(\theta)}^{l_1(\theta)} (D_1(\theta, l))^2 dl$$

$$= \frac{4}{15} a^5 \frac{\cos^3\theta}{\sin^2\theta} + \frac{4}{3} a^2 b^3 \sin\theta \qquad (18)$$

so that by (12),

$$E_{kk} = \frac{8}{\pi^2}\int_0^{\theta_3}\left(\frac{4}{15}b^5\frac{\sin^3\theta}{\cos^2\theta} + \frac{4}{3}a^3b^2\cos\theta\right)d\theta + \frac{8}{\pi^2}\int_{\theta_3}^{\pi/2}\left(\frac{4}{15}a^5\frac{\cos^3\theta}{\sin^2\theta} + \frac{4}{3}a^2b^3\sin\theta\right)d\theta$$

$$= \frac{8}{\pi^2}\left(\frac{8}{15}b^5\cos\theta_3 - \frac{8}{15}b^5 + \frac{4}{15}b^5\frac{\sin^2\theta_3}{\cos\theta_3} + \frac{4}{3}a^3b^2\sin\theta_3\right.$$

$$\left. + \frac{8}{15}a^5\sin\theta_3 - \frac{8}{15}a^5 + \frac{4}{15}a^5\frac{\cos^2\theta_3}{\sin\theta_3} + \frac{4}{3}a^2b^3\cos\theta_3\right)$$

When $\cos\theta_3 = \frac{b}{\sqrt{a^2+b^2}}$ and $\sin\theta_3 = \frac{a}{\sqrt{a^2+b^2}}$ are substituted into this expression, it reduces to

$$E_{kk} = \frac{64}{15\pi^2}\{(a^2+b^2)^{5/2} - a^5 - b^5\}\ .$$

Variance for a general rectangular pixel

The previous section computed the approximation E_{kk} to D_{kk} for a rectangular pixel R_k with center at the north pole O. If ithe center is at a general point $(x_0, y_0, 1)$ on the top face T of the hemicube, there are two modifications required. First of all, the factor cos $\alpha(\omega)$ $d\omega$ in (9) is no longer equivalent to $dx\ dy$. Instead, by a standard formula from [2] or [4],

$$\cos\alpha(\omega)\,d\omega = \frac{dx \cdot dy}{(1 + x^2 + y^2)^2} \tag{19}$$

We will assume that the pixel R_k is small enough so that the denominator of (19) can be evaluated at the pixel center (x_0, y_0) instead of at (x, y).

Our basic assumptions involve a probability distribution on the space S of great circles, and our second modification is to account for the map N which takes a line L in the space Q of lines on T to its corresponding great circle in S. Let $J(x, y, \theta, l)$ be the Jacobian "area stretching" determinant for this map. Then in analogy to (9) and (10)

$$D_{kk} = \frac{F(x_0, y_0)}{4\pi(1 + x_0^2 + y_0^2)^4} \tag{20}$$

where

$$F(x_0, y_0) = \frac{1}{\pi^2} \int_0^{2\pi} d\theta \int_{-\infty}^{\infty} dl \, (D(\theta, l))^2 J(x_0, y_0, \theta, l) \tag{21}$$

and $D(\theta, l)$ is defined in (11). The factor of $\frac{1}{4\pi}$ in (20) arises because a probability distribution must integrate to 1, while the solid angle measure for sets of normals in S integrates to 4π.

To compute $J(x, y, \theta, l)$, let $U(\theta, l)$ be the plane through $L(\theta, l)$ and $(0, 0, 0)$, and let $N(\theta, l)$ be its normal. A differential rectangle in Q with sides $d\theta$ and dl, and area $dA = d\theta \, dl$ is mapped to a differential parallelogram in S with sides $\frac{\partial N}{\partial \theta} d\theta$ and $\frac{\partial N}{\partial l} dl$, and area

$$d\omega = \left| \frac{\partial N}{\partial \theta} d\theta \times \frac{\partial N}{\partial l} dl \right|$$

so

$$J(x, y, \theta, l) = \frac{d\omega}{dA} = \left| \frac{\partial N}{\partial \theta} \times \frac{\partial N}{\partial l} \right| .$$

To proceed farther, we need a formula for $N(\theta, l)$ in terms of x, y, θ, and l. The plane $U(\theta, l)$ contains the vector $V_1 = (x + l \cos\theta, y + l \sin\theta, 1)$ from $(0, 0, 0)$ to the point G in figure 1, and also the vector V_2 = (- sinθ, cosθ, 0) in the direction of the arrow on L in figure 1. The right hand rule matches our orientation conventions, so

$$N(\theta, l) = \frac{V_1 \times V_2}{|V_1 \times V_2|}$$

$$= \frac{(-\cos\theta, -\sin\theta, x\cos\theta + y\sin\theta + l)}{\sqrt{1 + (x\cos\theta + y\sin\theta + l)^2}}$$

Then some standard calculus, algebra, and trigonometry can be used to derive

$$J(x, y, \theta, l) = \left|\frac{\partial N}{\partial\theta} \times \frac{\partial N}{\partial l}\right| = \frac{1}{1 + (x\cos\theta + y\sin\theta + l)^2}.$$

We will assume the pixels are small enough so that in the integral (21), l must be very small for $D(\theta, l)$ to be non-zero, and so $J(x, y, \theta, l)$ can be replaced by $J(x, y, \theta, 0)$. Once this is done, the same rotation and reflection symmetries apply as in (12), so, using (17) and (18),

$$F(x_0, y_0) = \frac{8}{\pi^2}\int_0^{\pi/2} d\theta \int_0^{\infty} dl\, (D(\theta, l))^2 J(x_0, y_0, \theta, 0)$$

$$= \frac{8}{\pi^2}\int_0^{\theta_3} d\theta \frac{4a^5\cos^3\theta\sin^{-2}\theta + 20a^2b^3\sin\theta}{15(1 + (x_0\cos\theta + y_0\sin\theta)^2)}$$

$$+ \frac{8}{\pi^2}\int_{\theta_3}^{\pi/2} d\theta \frac{4b^5\cos^{-2}\theta\sin^3\theta + 20a^3b^2\cos\theta}{15(1 + (x_0\cos\theta + y_0\sin\theta)^2)}.$$

For a pixel R_{kk} of the same shape with center $(1, y_0, z_0)$ on a side face of the hemicube, the formulas in [2] and [4] give

$$\cos\alpha\,(\omega)\,d\omega \;=\; \frac{z \cdot dy \cdot dz}{(1 + y^2 + z^2)^{\,2}}$$

so

$$D_{kk} \;=\; \frac{z_0^2 F\,(y_0, z_0)}{4\pi\,(1 + y_0^2 + z_0^2)^{\,4}}.$$

Correlation between pixel errors.

So far we have only considered the variance D_{kk} for a single pixel R_k. There are correlations D_{kl} between the errors on pairs of different pixels R_k and R_l which also contribute to the total form factor variance. To analyze these correlations, we again start with the case that both pixels are near the north pole of H, so that we can integrate with respect to area on the top hemicube face T.

Figure 5 shows two horizontally adjacent pixels, of width $2a$ and height $2b$, R_k on the left, with center $I = (- a, 0)$, and R_l on the right, with center $J = (a, 0)$. The origin $O = (0, 0)$ is at the midpoint of the common side NF of the two pixels. Also shown is the line $L(\theta, l)$ defined as in figure 1. Figures 5 through 9 show the five “general position” cases, which we will name by these figure numbers. Note that since F, J, and C lie on a straight line, if L intersects the interiors of segments OF and NC, the point J must lie above L, as shown in figure 5. Figures 5 through 9 have the same rotation and reflection symmetries as figure 1, so we again need only consider $0 \le \theta \le \pi/2$ and $l \ge 0$. (Note that the reversal of the orientation of $L(\theta, l)$ changes the sign of both D_k and D_l, but not their product.)

Figure 10 shows the regions in (θ, *l*) space corresponding to these cases, separated by the curves

$$m_0(\theta) = a \cos \theta$$

$$m_1(\theta) = 2a \cos \theta - b \sin \theta$$

$$m_2(\theta) = b \sin \theta$$

$$m_3(\theta) = b \sin \theta - 2a \cos \theta$$

at which the line $L(\theta, l)$, with equation $l = x \cos \theta + y \sin \theta$, passes through the points *J, C, F,* and *A* respectively. Key θ values where these curves intersect are also shown:

$$\theta_3 = \arctan(a/b)$$

$$\theta_4 = \arctan(2a/b)$$

and

$$\theta_5 = \arctan(3a/b).$$

Let

$$F_1 = \int_{-2a}^{0} dx \int_{-b}^{b} dy \, \{ V''(x, y) - V''(I) \}$$

and

$$F_2 = \int_{0}^{2a} dx \int_{-b}^{b} dy \, \{ V''(x, y) - V''(J) \}$$

where I = (-a, 0) and J = (a, 0) are the centers of R_k and R_l, respectively, as shown in figure 5. Then for cases 5, 6, and 7, F_1 is the area of triangle EFG, which works out to be

$$F_1 = \frac{(b\sin\theta - l)^2}{2\sin\theta\cos\theta},$$

and for cases 8 and 9, F_1 is the area of trapezoid $AFGK$,

$$F_1 = \frac{2a\,(b\sin\theta - a\cos\theta - l)}{\sin\theta}.$$

Also, for case 5, F_2 is the negative of the area of triangle GMN,

$$F_2 = -\frac{(b\sin\theta + l)^2}{2\sin\theta\cos\theta},$$

for cases 6 and 8, F_2 is the negative of the area of trapezoid $GHCN$,

$$F_2 = -\frac{2a\,(b\sin\theta - a\cos\theta + l)}{\sin\theta},$$

and for cases 7 and 9, F_2 is the area of trapezoid $FBHG$,

$$F_2 = \frac{2a\,(b\sin\theta + a\cos\theta - l)}{\sin\theta}.$$

The corresponding formulas for $D_i(\theta, l) = F_1 F_2$, the product of the errors on pixels R_k and R_l for case i, are

$$D_5(\theta, l) = -\frac{(b\sin\theta - l)^2\,(b\sin\theta + l)^2}{4\sin^2\theta\cos^2\theta},$$

$$D_6(\theta, l) = -\frac{a\,(b\sin\theta - l)^2\,(b\sin\theta - a\cos\theta + l)}{\sin^2\theta\cos\theta},$$

$$D_7(\theta, l) = \frac{a(b\sin\theta - l)^2 (b\sin\theta + a\cos\theta - l)}{\sin^2\theta\cos\theta},$$

$$D_8(\theta, l) = -\frac{4a^2 (b\sin\theta - a\cos\theta - l)(b\sin\theta - a\cos\theta + l)}{\sin^2\theta},$$

and

$$D_9(\theta, l) = \frac{4a^2 (b\sin\theta - a\cos\theta - l)(b\sin\theta + a\cos\theta - l)}{\sin^2\theta}.$$

Let

$$G(\theta) = \int_0^\infty D_i(\theta, l)\, dl.$$

where I is chosen according to the arrangement of cases shown in figure 10. Then $G(\theta)$ is

$$\int_0^{m_2(\theta)} D_5(\theta, l)\, dl, \qquad 0 \le \theta \le \theta_3,$$

$$\int_0^{m_1(\theta)} D_5(\theta, l)\, dl + \int_{m_1(\theta)}^{m_0(\theta)} D_6(\theta, l)\, dl + \int_{m_0(\theta)}^{m_2(\theta)} D_7(\theta, l)\, dl, \;\; \theta_3 \le \theta \le \theta_4,$$

$$\int_0^{m_3(\theta)} D_8(\theta, l)\, dl + \int_{m_3(\theta)}^{m_0(\theta)} D_6(\theta, l)\, dl + \int_{m_0(\theta)}^{m_2(\theta)} D_7(\theta, l)\, dl, \;\; \theta_4 \le \theta \le \theta_5,$$

$$\int_{0}^{m_0(\theta)} D_8(\theta, l)\, dl + \int_{m_0(\theta)}^{m_3(\theta)} D_9(\theta, l)\, dl + \int_{m_3(\theta)}^{m_2(\theta)} D_7(\theta, l)\, dl, \ \theta_5 \le \theta \le \pi/2.$$

Note that $D_5(\theta, l)$ through $D_9(\theta, l)$, when written with a common denominator $\cos^2\theta\sin^2\theta$, all have numerators whose terms are of total degree 4 in the variables $a\cos\theta$, $b\sin\theta$, and l. The functions $m_0(\theta)$ through $m_3(\theta)$ used as limits of integration are also linear combinations of $a\cos\theta$ and $b\sin\theta$. Integration by dl raises the power of l by 1, so the function $G(\theta)$ has numerator terms of total degree 5 and can be represented as

$$G(\theta) = \sum_{n=0}^{5} c_n a^{5-n} b^n \cos^{3-n}\theta \sin^{n-2}\theta \qquad (22)$$

where the coefficients c_n differ in each of the θ ranges listed above. Each of the terms can be integrated in θ in closed form, but the results will not be used here, because they apply only to pixels very close to the north pole of H. For a general pair of pixels on T, we must compute

$$D_{kl} = \frac{8}{4\pi^3(1+x^2+y^2)^4} \int_0^{\pi/2} \frac{G(\theta)\, d\theta}{1 + (x\cos\theta + y\sin\theta)^2},$$

and for a pair of pixels on a side face,

$$D_{kl} = \frac{8z^2}{4\pi^3(1+y^2+z^2)^4} \int_0^{\pi/2} \frac{G(\theta)\, d\theta}{1 + (y\cos\theta + z\sin\theta)^2}.$$

The case of two pixels sharing a horizontal edge instead of a vertical one is found from the case above by reflection in The 45° line $x = y$, as discussed for figure 3.

The case of two diagonally adjacent pixels is similar. Figure 11 shows two pixels of width $2a$ and height $2b$, touching at the origin O. The corresponding error integrals are

$$F_1 = \int_{-2a}^{0} dx \int_{0}^{2b} dy \{ V''(x, y) - V''(I) \}$$

and

$$F_1 = \int_{0}^{2a} dx \int_{-2b}^{0} dy \{ V''(x, y) - V''(J) \}$$

where $I = (-a, b)$ and $J = (a, -b)$. The five “general position” cases are shown as the lines L_{12}, L_{13}, L_{14}, L_{15}, and L_{16} in figure 11. Figure 12 shows the regions in (l, θ) space where each of these cases apply. These regions are separated by the curves

$$n_1(\theta) = 2b\sin\theta - 2a\cos\theta$$

$$n_2(\theta) = 2b\sin\theta$$

$$n_3(\theta) = 2a\cos\theta$$

$$n_4(\theta) = 2a\cos\theta - 2b\sin\theta$$

$$n_5(\theta) = b\sin\theta - a\cos\theta$$

and

$$n_6(\theta) = a\cos\theta - b\sin\theta,$$

defining lines which pass respectively through the points *A, B, E, F, I,* and *J* of figure 11. These curves intersect at the θ values θ_1 through θ_5, with

$$\theta_1 = \arctan(a/3b),$$

$$\theta_2 = \arctan(a/2b),$$

and θ_3, θ_4, and θ_5 as defined previously. As before, the integrals F_1 and F_2 can be found as areas of triangles and trapezoids. The product $F_1\,F_2$ can again be integrated with respect to l for fixed θ, according to the regions in figure 12 crossed by the vertical line at θ. The result is a separate formula $G(\theta)$ of the form of (22) for each of the six intervals $[\theta_i, \theta_{i+1}]$, where $\theta_0 = 0$ and $\theta_6 = \pi/2$. Then

$$D_{kl} = \frac{4}{4\pi^3 (1 + x^2 + y^2)^4} \int_0^{\pi/2} \frac{G(\theta)\, d\theta}{1 + (x\cos\theta + y\sin\theta)^2}.$$

Here there is only a factor of 4 in the numerator, because the reflections in the x and y axes are no longer symmetries of figure 11, only their product, a 180° rotation, is. However, the integral still extends only to $\pi/2$, since for $\pi/2 \le \theta \le \pi$, the line $L(\theta, l)$ intersects only one of the pixels R_k or R_l, so one of the terms in the product $F_1\,F_2$ is zero. An analogous formula holds for pixels on the side faces of the hemicube.

One can apply a similar analysis to pairs of pixels which are further apart, but I have not done so, because the contributions from the correlations D_{kl} for the 8 neighbors R_l surrounding pixel R_k already total less than 1% of the variance D_{kk} , and pixels farther apart will have even less correlation. In addition, if the separation between two pixels becomes too large, it may no longer be correct to assume that at most one polygon edge crosses between them, and that it does not have to be extended beyond its endpoints in order to intersect them. Nevertheless, geometric arguments like the ones above show that for any pair of nearby pixels R_k and R_l on T, the error correlation D_{kl} can be approximated by

$$D_{kl} = \frac{1}{4\pi^3 (1 + x^2 + y^2)^4} \sum_{j=0}^{J} \int_{\theta_j}^{\theta_{j+1}} d\theta \sum_{n=0}^{5} \frac{c_{jmn} a^{5-n} b^n \sin^{n-2}\theta \cos^{3-n}\theta}{1 + (x\cos\theta + y\sin\theta)^2}$$

where m is a configuration index depending on the relative position of pixels R_k and R_l , x and y are the coordinates of the point halfway between the pixel centers, and the division points θ_j depend only on the ratio of a and b. The coefficients c_{jmn} arise from integrals of $D(\theta, l)$ with respect to l, which were performed symbolically by Mathematica$^{\mathrm{TM}}$.

We will assume that the two pixels are close enough together so that the denominators can be evaluated at the center of pixel R_k , instead of at the midpoint O between them. Then we can group all terms D_{kl} (but not D_{lk}) for a fixed k, to get

$$W_k = \sum_l D_{kl} = \frac{a^5}{4\pi^3 (1 + x^2 + y^2)^4} \sum_{j=0}^{J} \int_{\theta_j}^{\theta_{j+1}} d\theta \sum_{n=0}^{5} \frac{d_{jn} r^n \sin^{n-2}\theta \cos^{3-n}\theta}{1 + (x\cos\theta + y\sin\theta)^2} \tag{23}$$

where $r = b/a$ and $d_{jn} = \sum_m c_{jmn}$. Analogous formulas hold for pixels on the side faces of the hemicube.

If we calculate the total error variance from the hemicube sampling by summing these terms W_k for all pixels R_k, we will neglect error correlations between nearby pixels on adjacent faces, and include correlations for certain potential neighbors which are actually beyond the edges of faces. However for increasing hemicube resolution, these “edge effects” become small.

Face variance as a Riemann integral

Suppose the top face T of the hemicube is divided into $2M$ square pixels of side $2a = 2b = 1/M$, so that $r = 1$. Then the sum of (23) over these pixels can be rewritten as

$$D_{top} = a^3 \sum_{l=-M}^{M-1} \sum_{m=-M}^{M-1} \frac{(2a)\,(2a)}{16\pi^3\,(1 + x_l^2 + y_m^2)^4} \sum_{j=0}^{J} \int_{\theta_j}^{\theta_{j+1}} d\theta \sum_{n=0}^{5} \frac{d_{jn} \sin^{n-2}\theta \cos^{3-n}\theta}{1 + (x_l \cos\theta + y_m \sin\theta)^2}$$

where $x_l = \frac{l + 0.5}{M}$ and $y_m = \frac{m + 0.5}{M}$. If we replace the $(2a)\ (2a)$ by $\Delta x\ \Delta y$ this becomes

$$D_{top} = a^3 \sum_{l=-M}^{M-1} \sum_{m=-M}^{M-1} e\,(x_l, y_m)\,\Delta x \Delta y \qquad (24)$$

where

$$e\,(x, y) = \frac{1}{16\pi^3\,(1 + x^2 + y^2)^4} \sum_{j=0}^{J} \int_{\theta_j}^{\theta_{j+1}} d\theta \sum_{n=0}^{5} \frac{d_{jn} \sin^{n-2}\theta \cos^{3-n}\theta}{1 + (x \cos\theta + y \sin\theta)^2}.$$

The double sum in (24) is a Riemann sum for a double integral, so when M approaches infinity in the limit of fine hemicube subdivision, the sum approaches the integral

$$T = \int_{-1}^{1} dx \int_{-1}^{1} dy \cdot e\,(x, y)\ ,$$

which is independent of the resolution of the hemicube.

The expression $e(x, y)$ is too complicated to be integrated analytically, so a sum is still required to estimate the integral. However, Simpson’s rule can be used, which gives a more accurate estimate than the Riemann sum (24). Note that because of the symmetry of the square, T can be computed from the integral over a fundamental triangle

$$T = 8 \int_{0}^{1} dy \int_{0}^{y} dx \cdot e\,(x, y)\ ,$$

requiring fewer terms in the Simpson’s rule sum. Similarly, the contribution from the four side faces is $D_{sides} = a^3 S$, where

$$S = 8\int_0^1 dz \int_0^1 dy \cdot f(y, z)$$

and

$$f(y, z) = \frac{z^2}{16\pi^3 (1 + y^2 + z^2)^4} \sum_{j=0}^{J} \int_{\theta_j}^{\theta_{j+1}} d\theta \sum_{n=0}^{5} \frac{d_{jn} \sin^{n-2}\theta \cos^{3-n}\theta}{1 + (y\cos\theta + z\sin\theta)^2}.$$

Optimization

Suppose we have a fixed number K of pixels, which are to be distributed over the top and sides of a hemicube. Even if all the pixels are square, we can use different resolutions for the top and the sides. So suppose we divide the top face into $2M$ by $2M$ pixels, with $a = 1/(2M)$, and each of the four side faces into $2N$ by N pixels, with $a = 1/(2N)$. Then

$$K = 4M^2 + 8N^2.$$

Let $u = 2M/\sqrt{K}$ and $v = 2N/\sqrt{K}$ be real-valued proxies for M and N, which we will use to apply calculus to the variance minimization. The constraint on total pixels becomes

$$u^2 + 2v^2 = 1,$$

and we can write the total variance of D as

$$E\left(D^2\right) = D_{top} + D_{sides} = \left(\frac{1}{2M}\right)^3 T + \left(\frac{1}{2N}\right)^3 S$$

$$= \frac{1}{K^{3/2}}\left(\frac{T}{u^3} + \frac{S}{v^3}\right).$$

Let $t = v^2$ so that $v = t^{1/2}$ and $u = (1\text{-}2t)^{1/2}$. Then we must minimize

$$w(t) = \frac{T}{(1-2t)^{3/2}} + \frac{S}{t^{3/2}}$$

Setting the derivative equal to zero, we get

$$\frac{dw}{dt} = -\frac{3}{2}\left(-2\frac{T}{(1-2t)^{5/2}} + \frac{S}{t^{5/2}}\right) = 0\,.$$

Solving for t, one finds

$$t = \frac{S^{2/5}}{(2T)^{2/5} + 2S^{2/5}}.$$

Using this t and the definitions of u, v, and t, we can find the integer resolutions

$$M = \sqrt{(1-2t)}K/2 \qquad (25)$$

and

$$N = \sqrt{t}K/2\,, \qquad (26)$$

where the "=" signs imply truncation of the fractional part. With these minimizing t, u, and v, the total variance reduces, after some algebraic manipulation, to

$$E\left(D^2\right) = K^{-3/2}(T^{2/5} + 2^{3/5}S^{2/5})^{5/2}\,.$$

Since S and T are constant, this is proportional to $K^{-3/2}$. As discussed above, the variance of the mean of K independent measurements decreases only as K^{-1}. The improvement in convergence here comes because the measurements are locally correlated, and the regular sampling takes advantage of this correlation, in a way that random samples cannot.

We now investigate several ways of improving this optimum if the scan conversion onto the hemicube can be more flexible. The first is to replace the cube by a rectangular solid. Because of the four-fold symmetry in the horizontal plane, the horizontal cross-section will still be a square of side 2, but the vertical height h can be different than 1. In order to apply the previous analysis to the top face, we project it onto the horizontal plane tangent to the unit sphere, getting a square of half-width $s = 1/h$. Let K, M, N, u, v, and t be as before, except that the side faces are now hN pixels high. We can then repeat the above analysis for square pixels, using

$$T_h = 8s^3 \int_0^s dy \int_0^y dx \cdot e(x, y)$$

and

$$S_h = 8 \int_0^h dz \int_0^1 dy \cdot f(y, z) .$$

The reason for the s^3 factor in front of the integral for T_h is that the half-width of each pixel becomes $s/(2M)$. The total pixel count is now

$$K = 4M^2 + 8hN^2$$

so the constraint equation in u and v becomes

$$u^2 + 2hv^2 = 1$$

and (25) must be replaced by

$$M = \sqrt{(1 - 2ht) K}/2 .$$

The minimum variance is at

$$t = \frac{S_h^{2/5}}{(2hT_h)^{2/5} + 2hS_h^{2/5}}$$

with value

$$E\left(D^2\right) = K^{-3/2}\left(T_h^{2/5} + (2h)^{3/5}S_h^{2/5}\right)^{5/2}.$$

The next generalization is to use non-square pixels. Most scan-conversion hardware can deal with rectangular pixels by adjusting the 4 by 4 viewing projection matrix, as long as all the rectangles are identical, and are arranged in a lattice. By four-fold symmetry, the optimal lattice on the top must still be a square one, but on the four sides, the optimal lattice is truly rectangular. Let *r* be the ratio of the height to the width of the rectangular pixels in the four sides. Again take *K, M, N, u, v,* and *t* as before, except that the side faces are now *hN/r* rectangular pixels high. Then we modify the formula for S_h to

$$S_{hr} = \frac{8}{r}\int_0^h dz \int_0^1 dy \cdot g(y, z, r)$$

where, including *r* as in (23),

$$g(y, z, r) = \frac{z^2}{16\pi(1 + y^2 + z^2)^4} \sum_{j=0}^{J} \int_{\theta_j}^{\theta_{j+1}} d\theta \sum_{n=0}^{5} \frac{r^n d_{jn} \sin^{n-2}\theta \cos^{3-n}\theta}{1 + (y\cos\theta + z\sin\theta)^2}.$$

The extra factor of 1/*r* in front of the integral for S_{hr} arises because the Δz in the Riemann sum needs to be *r/N* instead of 1/*N*. The new constraint equation becomes

$$u^2 + (2h/r)v^2 = 1.$$

The minimum variance is at

$$t = \frac{S_{hr}^{2/5}}{((2h/r)T_h)^{2/5} + (2h/r)S_{hr}^{2/5}}$$

with value

$$E\left(D^2\right) = K^{-3/2}(T_h^{2/5} + (2h/r)^{3/5}S_{hr}^{2/5})^{5/2}.$$

Appendix A describes hemicubes with uneven pixel grid spacing, specified by low degree polynomials, and Appendix B describes how to modify the standard scan conversion algorithm to accommodate such uneven grids.

Results

The total variance was minimized in each of the cases described above, using an unconstrained optimizer written by David Gay [19]. It had to estimate the gradient of the variance by finite differences, since it was impossible to differentiate the variance analytically with respect to the pixel and hemicube size and shape parameters.The variance depends on the the scene geometry and the number *K* of pixels used, so I will express the results as a ratio of the optimum variance obtained to the variance from a standard hemicube of sides 2 x 2 x 1, with the same number *K* of equal square pixels. This base case is line 1 in table 1, which presents the other cases in the order discussed below.

The first optimization was for a standard 2 x 2 x 1 hemicube with square pixels, but allowing more smaller pixels on the top than on the sides. The optimal value of *t* to use in (25) and (26) is .238126, and the variance ratio is .75809. The next optimization was to allow the side faces of the hemicube to have height different than 1, but still have square pixels. In this case the optimal height *h* of the sides was 1.41647, and the optimum *t* was .232992, giving a variance ratio of .68614. Note that the side faces are now 2*N* by *hN* pixels, where *N* is determined from (26). This optimum made the hemicube taller than half a cube, the opposite distortion to that in Recker *et al*. [9], whose goal was good early approximations in progressive radiosity.

If in addition the sides were allowed to have identical rectangular pixels, the optimum ratio *r* of the height to the width of these rectangles was 1.0959, the hemicube height *h* was 1.42054, the value of *t* was .255042, and the variance ratio was .68554. Now the side faces are *hN/r* pixels high, and (25) must be modified accordingly.

	variance ratio	***t***	***r***	***x*(*u*)**	***y*(*u*)**	***z*(*v*)**
1	1.0	0.333333	1.0	u	u	v
2	0.75809	0.238126	1.0	u	u	v
3	0.68614	0.232992	1.0	$0.70598\ u$	u	$1.41647\ v$
4	0.68554	0.255042	1.09590	$0.70396\ u$	u	$1.42054\ v$
5	0.60711	0.271831	1.0	$0.56563\ u + 0.33410\ u^2$	$0.60746\ u + 0.39254\ u^2$	$1.23056\ v - 0.11911\ v^2$
6	0.60583	0.245522	0.85616	$0.55342\ u + 0.29327\ u^2$	$0.61429\ u + 0.38572\ u^2$	$1.26946\ v - 0.08839\ v^2$
7	0.56897	0.270982	1.0	$0.69628\ u - 0.15892\ u^2 + 0.35680\ u^3$	$0.76101\ u - 0.17971\ u^2 + 0.41870\ u^3$	$1.69337\ v - 1.44387\ v^2 + 0.86886\ v^3$
8	0.56884	0.248993	0.81477	$0.62711\ u - 0.09508\ u^2 + 0.24086\ u^3$	$0.76554\ u - 0.16232\ u^2 + 0.39678\ u^3$	$1.86719\ v - 1.67351\ v^2 + 1.10017\ v^3$
9	0.62488	0.17545	1.0	$0.95310\ u - 0.83551\ u^2 + 1.34289\ u^3$	$0.74430\ u - 0.34758\ u^2 + 0.60328\ u^3$	$1.37418\ v - 1.28992\ v^2 + 0.60044\ v^3$

Table 1: Hemicube parameters. The rows are 1) standard hemicube, 2) unequal resolutions on the top and side, 3) 2x2x*h* hemicube, 4) 2x2x*h* hemicube with rectangular pixels, 5) uneven quadratic spacing, 6) quadratic spacing with variable *r*, 7) uneven cubic spacing, 8) uneven cubic spacing with variable *r*, and 9) uneven cubic spacing optimizing equal contribution to the form factor.

For the case of uneven spacing described in Appendix A, I tried quadratic and cubic polynomials on the top and the side faces. For example, the cubic polynomials have 7 independent parameters: a_1, a_2, a_3, b_1, b_2, c_1, and c_2. The size s of the top is determined by (28)in Appendix A, and then b_3 and c_3 are determined from (29) and (30) respectively, using $h = 1/s$. For the top face the optimum polynomial was

$$x(u) = 0.696285u - 0.158924u^2 + 0.356801u^3,$$

and for the side faces, the optimum polynomials were

$$y(u) = 0.761007u - 0.179706u^2 + 0.418699u^3$$

and

$$z(v) = 1.69337v - 1.44387v^2 + 0.868916v^3.$$

The value of t was .270982, and the variance ratio was .56897. Note that (30) assures that the sides have the correct height h even when the v range [0, 1] is divided into N equal parts.

To allow flexibility in the ratio of the vertical to horizontal pixel counts on the side faces, suppose that there are N/r vertical subdivisions of the v interval [0, 1], with the usual $2N$ subdivisions of the u interval [-1, 1]. Then r becomes an eighth variable in the minimization. Its optimal value is .814768, and the optimal polynomials are shown on row 8 of table 1. The value of t is .248993, and the variance ratio is .56884, a 43% improvement over the standard hemicube with square pixels, all of the same size. Note that a 31% improvement, or three quarters of the 43% above, can be achieved just by changing the shape and relative resolution of the top and sides of the hemicube, which is easy using current hardware.

Note also in table 1 that the variance improvements on lines 4, 6, and 8, from letting r be different from 1, are all insignificant. Figure 13 shows the hemicubes resulting from the cases in rows 2, 3, 5, and 7 of table 1. The first column shows the top face, in the size it would appear when projected onto the plane z = 1. The second column shows a side face. The last column shows the assembled hemi-solid, with the top face expanded to size 2 x 2. The number K of pixels allowed was set to 1000. The second row used 940 pixels, and the other three rows used 996.

Verification

To verify the performance of the proposed hemicube schemes, two independent tests were performed. These tests are reported in greater detail in Max and Troutman [20], which also describes a test on a “Cornell Room”. The first used 10,000 random triangles in each of the cases in figure 13, and computed the form factors within the finite element radiosity system used to produce figure 6 of [20]. For the standard hemicube, used as reference for the variance ratios, 40,000 triangles were used. The specified pixel counts *K* were from 25,000 to 50,000 increasing in steps of 5000. The random number generator was seeded from the clock, so that different triangles were generated for each test.

The second test used 20,000 random triangles for each case, in an independently coded program for computing form factors only. The specified pixel counts were from 3888 to 1,920,000 in 9 doubling steps, so that the final count corresponded to a 800 x 800 x 400 resolution hemicube. The random number generator was reinitialized for each case and resolution, so that the same random triangles were used.

Table 2 shows the results of the two tests. The row numbers correspond to those in table 1. The last two columns give the slopes of the least squares fit lines. In order to get a single number for the variance ratios reported in the middle two columns, the data for each case was fit with an enforced slope of -1.5, and the variance ratio reported is the antilog of the difference in the y-intercepts between the listed and standard cases. The slopes for test

2 are close to the predicted value of -1.5, and the variance ratios for both tests are also close to those predicted, but a little larger in all cases.

type	**predicted variance ratio**	**test 1 variance ratio**	**test 2 variance ratio**	**test 1 slope**	**test 2 slope**
1. standard	1.0	1.0	1.0	-1.47268	-1.52282
2. unequal	0.75809	0.78449	0.79526	-1.37623	-1.51190
3. rectangular	0.68614	0.72077	0.69620	-1.42348	-1.51731
5. quadratic	0.60711	0.66302	0.64671	-1.37450	-1.50050
7. cubic	0.56897	0.64006	0.58415	-1.42543	-1.51604
9. equal FFs	0.62488		0.63435		-1.52964

Table 2. Performance test results. Rows are numbered as in table 1.

One of the anonymous TVCG reviewers requested a comparison with the “equal contribution to the form factor” method of Sillion and Puech [8]. To do this, I minimized the mean square deviation of the pixel form factors from the mean pixel form factor, using cubic polynomial spacing as on line 7 of table 1. The extra degree of freedom *r* on line 8, affecting the pixel shape on the side faces, has no meaning here, since the form factor for a small region depends only on its size, and not on its shape. By the methods described above, the summed square deviation was interpreted as a Riemann integral, accurately approximated by Simpson’s rule, and minimized with respect to the independent cubic polynomial coefficients.

The resulting cubic polynomials are given in row 9 of table 1. The optimal hemicube was short and wide, as shown in figure 14, with a height *h* of only 0.68471, and a *t* value of 0.17545. The predicted improvement was 37.5%, not as good as the 43% in lines 7 and 8. This inferior performance for the same degrees of freedom verifies that the optimization criterion used in this paper gives better performance than the “equal contribution to the form factor" criterion. However the larger negative slope in the last column means that this method caught up to the one in row 7 at high hemicube resolution.

Future work

The "second basic assumption," about randomness of edges, does not apply to architectural scenes, where edges and surface normals are directed preferentially along the *x*, *y*, and *z* axes. It is then essential to rotate the hemicube around its "vertical" axis, in order to avoid positive error correlation in whole rows and columns of pixels along projected edges. Once this is done, the two families of "horizontal" edges become randomized.

The vertical edges parallel to the normal remain vertical, with their projection planes all passing through the north pole O of the hemicube. On the top face of the hemicube, such edges project to lines radiating from O. Therefore they result in lines $L(\theta, l)$ crossing a pixel at (x, y) with $\theta = \tan^{-1} (y/x)$. We already know how to compute the variance from such lines, so we can just skip the step of integrating over θ, to get a special variance for them. Similarly, their projections on the side faces are all vertical, so it is easy to get a special variance for the sides, which must now include the correlation between distant pixels in the column containing the edge, leading to a large positive correlation W_k in equation (23). Then we can use a weighted sum of the usual and special variances, to account for the proportion of edges parallel to the normal. For example, if all object edges are axis-aligned, an expected 1/3 of them will be vertical.

If this combined variance were minimized, an optimal hemicube could be designed for this special distribution of geometries. I suspect it would have a wider top, and shorter sides with higher horizontal resolution, corresponding to $h < 1$ and $r > 1$ in the notation here, in order to minimize the large error correlation discussed above. It would be better to use a rotated coordinate grid on the side faces, to break up this correlation.

Various authors, for example Beran-Koehn and Pavicic [21], have proposed using faces tilted at different angles, and perhaps a different number of them. It should also be possible to optimize over face orientation angles as well as face grids.

Peter Shirley has suggested that for shooting methods of progressive radiosity, the variance in radiosity from one shot is proportional to the form factor variance times the “unshot power”, so one can also use this analysis to dynamically change the number K of total samples based on the power to be shot. More generally, for ray traced samples, there need be no pattern compatible with scan-conversion, so all the sample directions could be independent parameters to optimize. However, this could result in a huge number of variables. In addition, the region corresponding to a ray would become the Voronois spherical polygon of directions closer to that ray than to any other, which would greatly complicate the geometrical analysis.

Dippe and Wold [22] propose using hexagonal grids, where each pixel is surrounded by six neighbors instead of four. Uniform hexagonal lattices are compatible with scan-conversion hardware, using a shearing viewing transformation to take them to a standard square lattice. I tested hexagonal versions of each of the cases in table 2, using the same parameters as in table 1, and the same 20,000 triangles described in test 2 above, and found no improvement in the variance. Hexagonal grids may be superior for image reconstruction from samples, as suggested by their fourier transforms, but do not seem to be better for estimating form factors. The mathematical analysis in this paper could be repeated for hexagonal grids, to help resolve this question.

Acknowledgments

Ned Greene suggested unequal resolution faces and unevenly spaced rows and columns of pixels as potential improvements for hemicube sampling. Fred Fritch helped me obtain the algorithm [19] for unconstrained optimization. Charles Grant and anonymous

ACM and IEEE reviewers provided useful suggestions for improvement. Roy Troutman performed the first test described in the Verification section. This work was performed under the auspices of the U. S. Department of Energy by Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

References

[1] Cindy Goral, Kenneth Torrance, Donald Greenberg, and Bennett Battaile, “Modeling the interaction of light between diffuse surfaces”, Computer Graphics Vol. 8 No. 3 (1984, Siggraph ‘84 proceedings) pp. 213 - 222.

[2] Michael Cohen and Donald Greenberg, “The hemi-cube: a radiosity solution for complex environments”, Computer Graphics Vol. 19 No. 3 (1985, Siggraph ‘85 proceedings) pp. 31 - 40.

[3] Michael Cohen, Shenchang Eric Chen, John Wallace, and Donald Greenberg, “A progressive refinement approach to fast radiosity image generation”, Computer Graphics Vol. 22 No. 4 (1988, Siggraph ‘88 proceedings) pp. 75 - 84.

[4] Robert Siegal and John Howell, “Thermal Radiation Heat Transfer,” Third Edition, Hemisphere Publishing Corporation, Washington (1992).

[5] Tomoyuki Nishita and Eihachiro Nakamae, “Continuous-tone representation of three dimensional objects taking into account of shadows and interreflection”, Computer Graphics Vol. 22 No. 4 (1985, Siggraph ‘85 proceedings) pp. 23 - 30.

[6] Daniel Baum, Holly Rushmeier, and James Winget “Improving radiosity solutions through the use of analytically determined form-factors”, Computer Graphics Vol. 23 No. 3 (1989, Siggraph ‘89 Proceedings) pp. 325 - 334.

[7] Peter Schröder and Pat Hanrahan “On the Form Factor between Two Polygons,” Computer Graphics (Annual Conference Series 1993) pp. 163 - 164.

[8] Francois Sillion and Claude Peuch, “A general two-pass method integrating specular and diffuse reflection”, Computer Graphics Vol. 23 No. 3 (1989, Siggraph ‘89 proceedings) pp. 335 - 344.

[9] Rodney Recker, David George, and Donald Greenberg, “Acceleration Techniques for Progressive Refinement Radiosity,” Computer Graphics Vol. 24 No. 2 (March 1990) pp. 59 - 66.

[10] John Wallace, Kells Elmquist, and Eric Haines, “A ray tracing algorithm for progressive radiosity”, Computer Graphics Vol. 23 No. 3 (1989, Siggraph ‘89 proceedings) pp. 315 - 324.

[11] Turner Whitted, “An Improved Illumination Model for Shaded Display,” Communications of the ACM Vol. 23 No. 6 (June 1980) pp. 343 - 349.

[12] Mark Lee, Richard Redner, and Sam Uselton, “Statistically Optimized Sampling for Distributed Ray Tracing,” Computer Graphics Vol. 19 No. 3 (July 1985) pp. 61 - 67.

[13] David Burnett, “Finite Element Analysis,” Addison Wesley, Reading Mass. (1988).

[14] Harold Zatz, “Galerkin Radiosity: A Higher Order Solution Method for Global Illumination,” Computer Graphics (Annual Conference Series 1993) pp. 213 - 220.

[15] Nelson Max and Michael Allison, “Linear radiosity approximation using vertex-to-vertex form factors”, in Graphics Gems III, David Kirk, editor, Academic Press, San Diego (1992) pp. 318 - 323.

[16] Roy Troutman and Nelson Max, “Radiosity Algorithms using Higher Order Finite Elements Methods,” Computer Graphics (Annual Conference Series 1993) pp. 209 - 212.

[17] Steven Gortler, Peter Schröder, Michael Cohen, and Pat Hanrahan, “Wavelet Radiosity,” Computer Graphics (Annual Conference Series 1993) pp. 221 - 230.

[18] Shenchang Eric Chen, Holly Rushmeier, Gavin Miller, and Douglas Turner, “A progressive multi-pass method for global illumination”, Computer Graphics Vol. 25 No. 4 (1991, Siggraph ‘91 proceedings) pp. 165 - 174.

[19] David Gay, Algorithm 611, Collected algorithms from the ACM, also in ACM Transactions on Mathematical Software Vol. 9, No. 4 (1983) pp. 503 - 524.

[20] Nelson Max and Roy Troutman, “Optimal Hemicube Sampling,” Proceedings of the Fourth Eurographics Workshop on Rendering, Eurographics Technical Report EG 93 RW, ISSN 1017-4656, pp. 185 - 200, and Addendum, pp. 348 - 351.

[21] Jeffery Beran-Koehn and Mark Pavicic, “A Cubic Tetrahedral Adaptation of the Hemi-Cube Algorithm,” in “Graphic Gems II,” James Arvo, editor, Academic Press, Boston (1991) pp. 299 - 302.

[22] Mark Dippe and Erling Wold, “Antialiasing Through Stochastic Sampling,” Computer Graphics Vol. 19 No. 3 (July 1985) pp. 69 - 78.

Appendix A: Uneven grids

A final generalization is to allow unevenly spaced pixels, which are impossible for most scan-conversion hardware. However, if rectangular pixels are arranged in straight rectilinear rows and columns, Appendix B shows how to modify software scan-conversion routines to deal with uneven row and column spacing, while still taking advantage of area coherence. (Presumably microcode or hardware could also be so modified.)

In order to define the uneven spacing by a few parameters which are independent of the resolution, we use polynomials of degree *deg*, so that the parameters are the coefficients. Suppose *u* and *v* are variables which range from -1 to 1, and are to be divided evenly, so that, for example $u_l = (l + .5)/M$. On the top face, by symmetry, the optimal spacing in *x* and *y* will be the same, so they are defined by identical polynomials

$$x(u) = \sum_{n=1}^{deg} a_n u^n$$

and

$$y(v) = \sum_{n=1}^{deg} a_n v^n \ ,$$

for positive *u* and *v*, extended for negative *u* and *v* to become odd functions, *i.e.*, $x(-u) = -x(u)$. The pixel with index (l, m) extends in *x* from $x(l/M)$ to $x((l+1)/M)$, so it has width approximately $\Delta x = x'((l+.5)/M)/M$ and height $\Delta y = y'((m+.5)/M)/M$, with the primes denoting differentiation with respect to *u* or *v*. Then the formula for D_{top} becomes

$$D_{top} = \frac{1}{M^3} \sum_{l=-M}^{M-1} \sum_{m=-M}^{M-1} e(u_l, v_m) \Delta u \Delta v \qquad (27)$$

where

$$e(u, v) = \frac{1}{16\pi^3 (1 + x(u)^2 + y(v)^2)^4} \sum_{j=0}^{J} \int_{\theta_j}^{\theta_{j+1}} d\theta \sum_{n=0}^{5} \frac{x'(u)^{5-n} y'(v)^n d_{jn} \sin^{n-2}\theta \cos^{3-n}\theta}{1 + (x(u)\cos\theta + y(v)\sin\theta)^2}$$

In the limit for large *M*, the Riemann sum (27) approaches

$$T = \int_{-1}^{1} dv \int_{-1}^{1} du \cdot e(u, v)$$

$$= 8\int_{0}^{1} dv \int_{0}^{v} du \cdot e(u, v)$$

as before. Note that the half-width *s* of the top face is now

$$s = \sum_{n=1}^{deg} a_n \ . \tag{28}$$

A similar calculation holds for the side faces, but now there is no symmetry in *y* and *z* so there are two different polynomials

$$y(u) = \sum_{n=1}^{deg} b_n u^n$$

and

$$z(v) = \sum_{n=1}^{deg} c_n v^n \ .$$

Of the *deg* coefficients in each of these two polynomials, only *deg* - 1 are independent parameters to optimize, because *y*(1) = 1, so

$$\sum_{n=1}^{deg} b_n = 1 \tag{29}$$

and *z*(1) = *h* = 1/*s*, the height of the side faces, so

$$\sum_{n=1}^{deg} c_n = \frac{1}{s} \ . \tag{30}$$

Appendix B: Scan conversion for uneven pixel spacing

To use the polynomials, the usual scan conversion algorithm must be modified for the uneven pixel spacing. For one face of the hemicube, let $x(i)$ and $y(j)$ be the tabulated column and row spacing for the pixel centers, and also tabulate $\Delta x(i) = x(i+1) - x(i)$ and $\Delta y(i) = y(i+1) - y(i)$. Subroutines are required for calculating **firsti**$(x) = \min\{i \mid x(i) \geq x\}$ and **firstj**$(y) = \min\{j \mid y(j) \geq y\}$. For quadratic polynomials, they can be implemented using the quadratic formula. For more general spacing, estimate **firsti** from a table for the inverse function of $x(i)$, and then verify and possibly adjust it using the table for $x(i)$.

Place the edges of the polygon in y-buckets, based on **firstj** of their minimum endpoint y. Then simplified pseudo-code for scan conversion in increasing y is:

```
For j = 0 to jmax do
  Insert edges from y-bucket(j) into x-sorted list
  While x-sorted list is non-empty
    Remove a pair of edges (edgel,edger)
    dz = (edger.z - edgel.z) / (edger.x - edgel.x)
    il = firsti(edgel.x)
    ir = firsti(edger.x)
    z = (il - edgel.x)*dz
    For i = il to ir-1 do
      If z is closer than zbuffer(i,j)
        zbuffer(i,j) = z
        itembuffer(i,j) = polygonID
      z = z + dz*Δx(i)
  For all edges in x-sorted list
    If edge ends on current scan line, remove it
    else update edge.x and edge.z using Δy(j).
```

Compared to the standard algorithm, the innermost loop requires an extra table access for $\Delta x(i)$, and one extra multiplication by $\Delta x(i)$.

Figure Captions

Fig. 1: Geometry when $\max(l_2,l_3) \le l \le l_1$.

Fig. 2: Geometry when $\theta \ge \theta_3$ and $0 \le l \le l_2(\theta)$.

Fig. 3: Geometry when $\theta \le \theta_3$ and $0 \le l \le l_3(\theta)$.

Fig. 4: The (θ, l) ranges for cases 1, 2, and 3.

Fig. 5: Case 5.

Fig. 6: Case 6.

Fig. 7: Case 7.

Fig 8: Case 8.

Fig 9: Case 9.

Fig. 10: The (θ, l) ranges for cases 5 through 9.

Fig. 11: Lines in cases 12 through 16.

Fig. 12: The (θ, l) ranges for cases 12 through 16.

Fig. 13: Grids on four kinds of optimized hemicubes. The rows are: 1) unequal resolutions on the top and side, 2) 2x2xh hemicube, 3) uneven quadratic spacing, 4) uneven cubic spacing.

Fig. 14: Grid on hemicube making pixel form factors as equal as possible.

Biography

Nelson Max is Professor of Applied Science at the University of California Davis, and a Computer Scientist at Lawrence Livermore National Laboratory. He received a Ph. D. in mathematics from Harvard University in 1967. He has taught mathematics and computer science at UC Berkeley, the University of Georgia, Carnegie Mellon University, and Case Western Reserve University. He was director of the NSF supported Topology Films Project in the early 1970's, which produced computer animated educational films on mathematics. He has worked in Japan for 3 and a half years as co-director of two Omnimax (hemisphere screen) stereo films for international expositions, showing the molecular basis of life. His computer animation has won numerous awards. His research interests are realistic computer rendering including shadow effects and radiosity, scientific visualization, volume and flow visualization, molecular modeling, and computer animation.

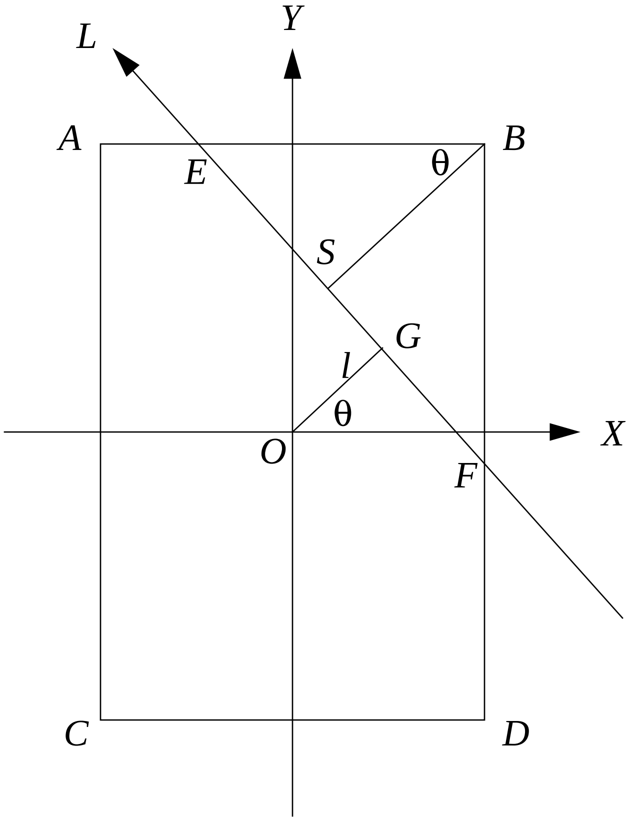

Fig. 1: Geometry when max(l_2,l_3) ≤ l ≤ l_1.

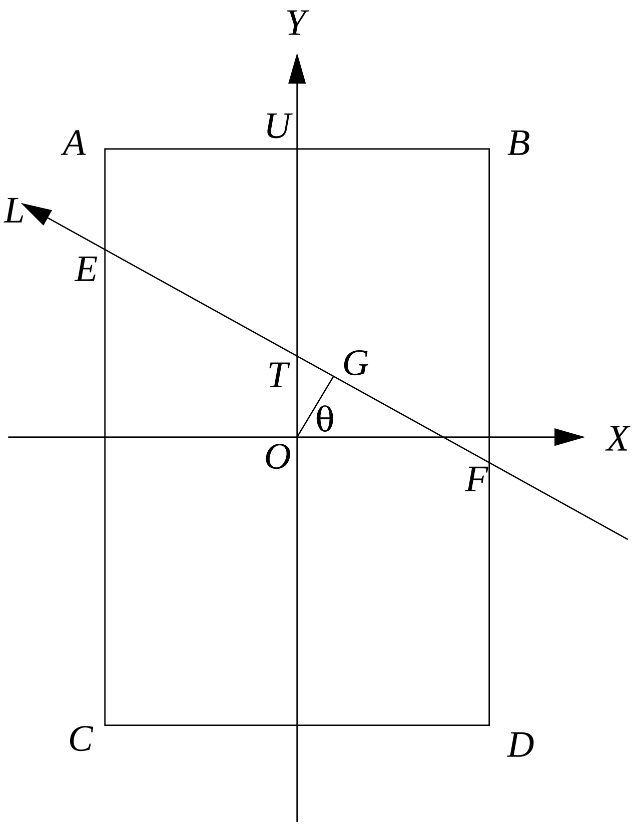

Fig. 2: Geometry when $\theta \geq \theta_3$ and $0 \leq l \leq l_2(\theta)$.

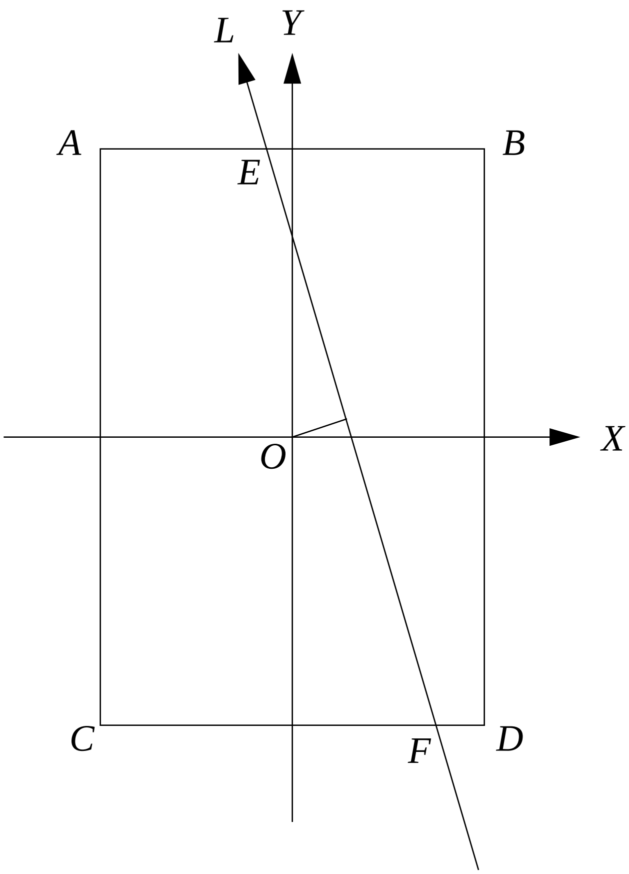

Fig. 3: Geometry when $\theta \leq \theta_3$ and $0 \leq l \leq l_3(\theta)$.

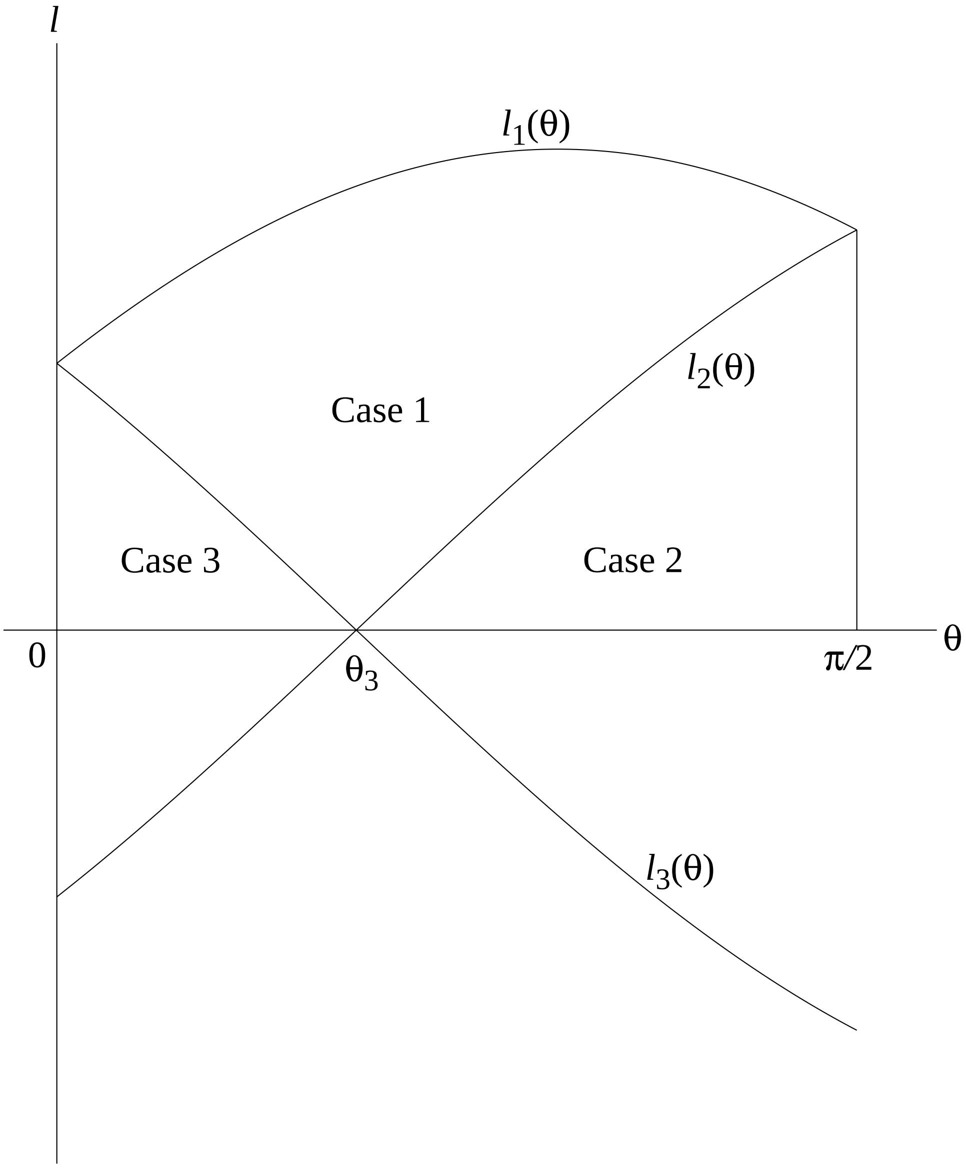

Fig. 4: The (θ, l) ranges for cases 1, 2, and 3.

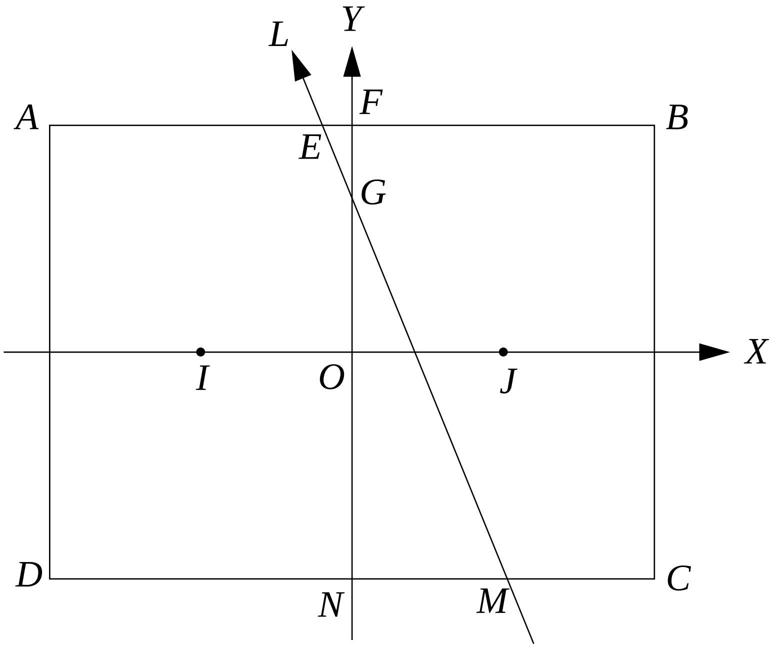

Fig. 5: Case 5.

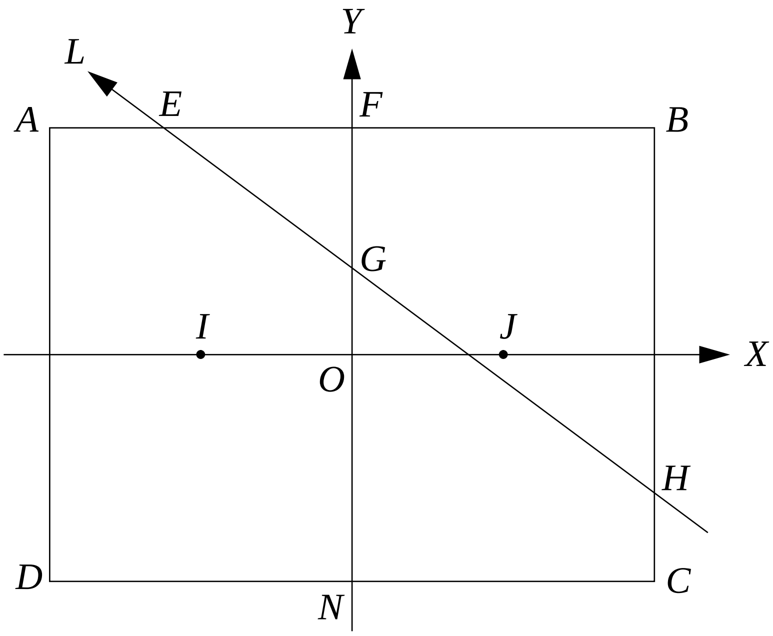

Fig. 6: Case 6.

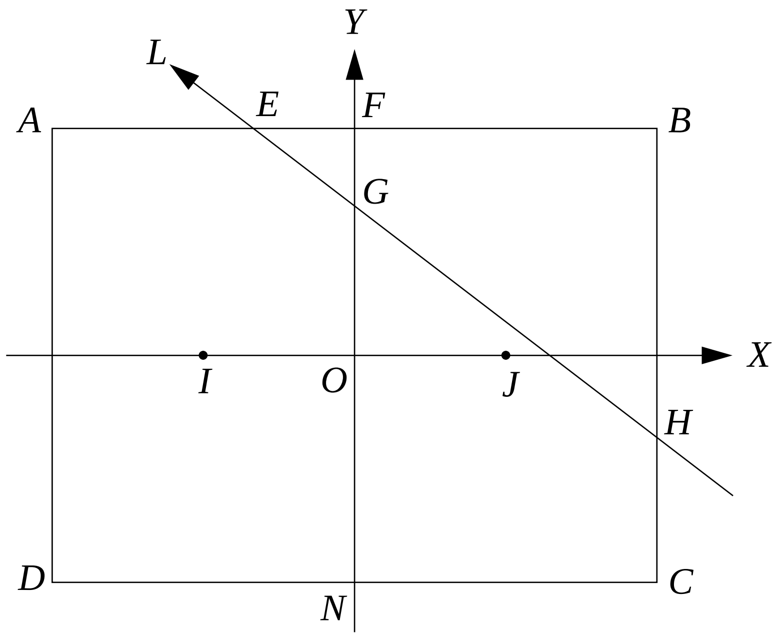

Fig. 7: Case 7.

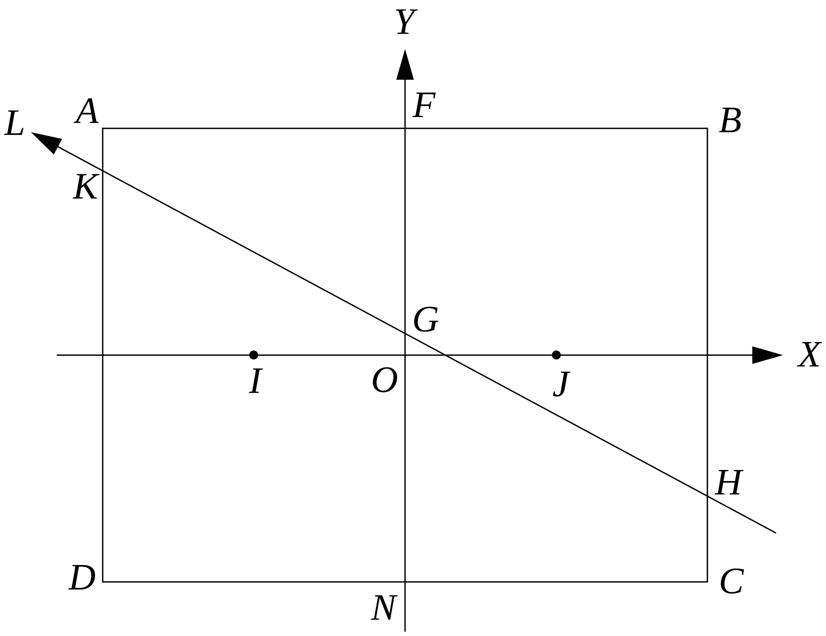

Fig 8: Case 8.

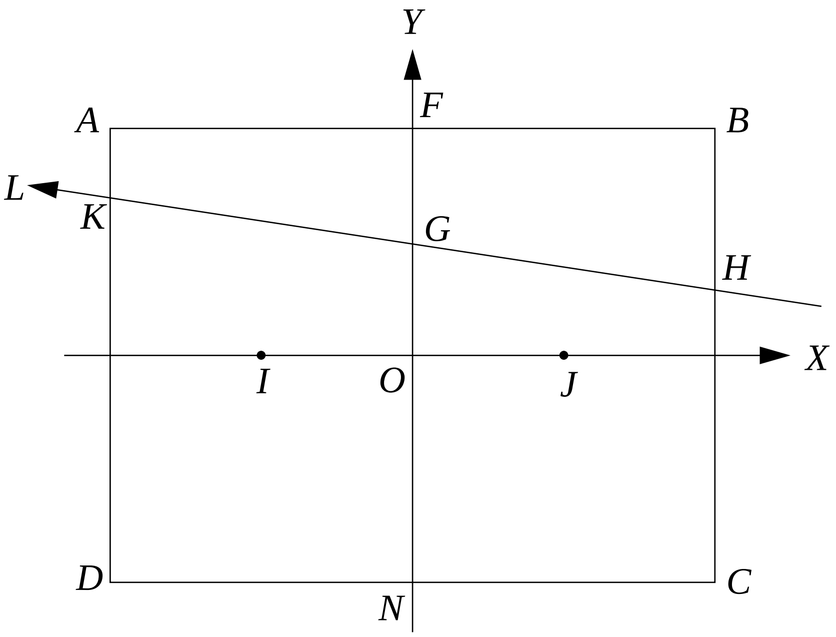

Fig 9: Case 9.

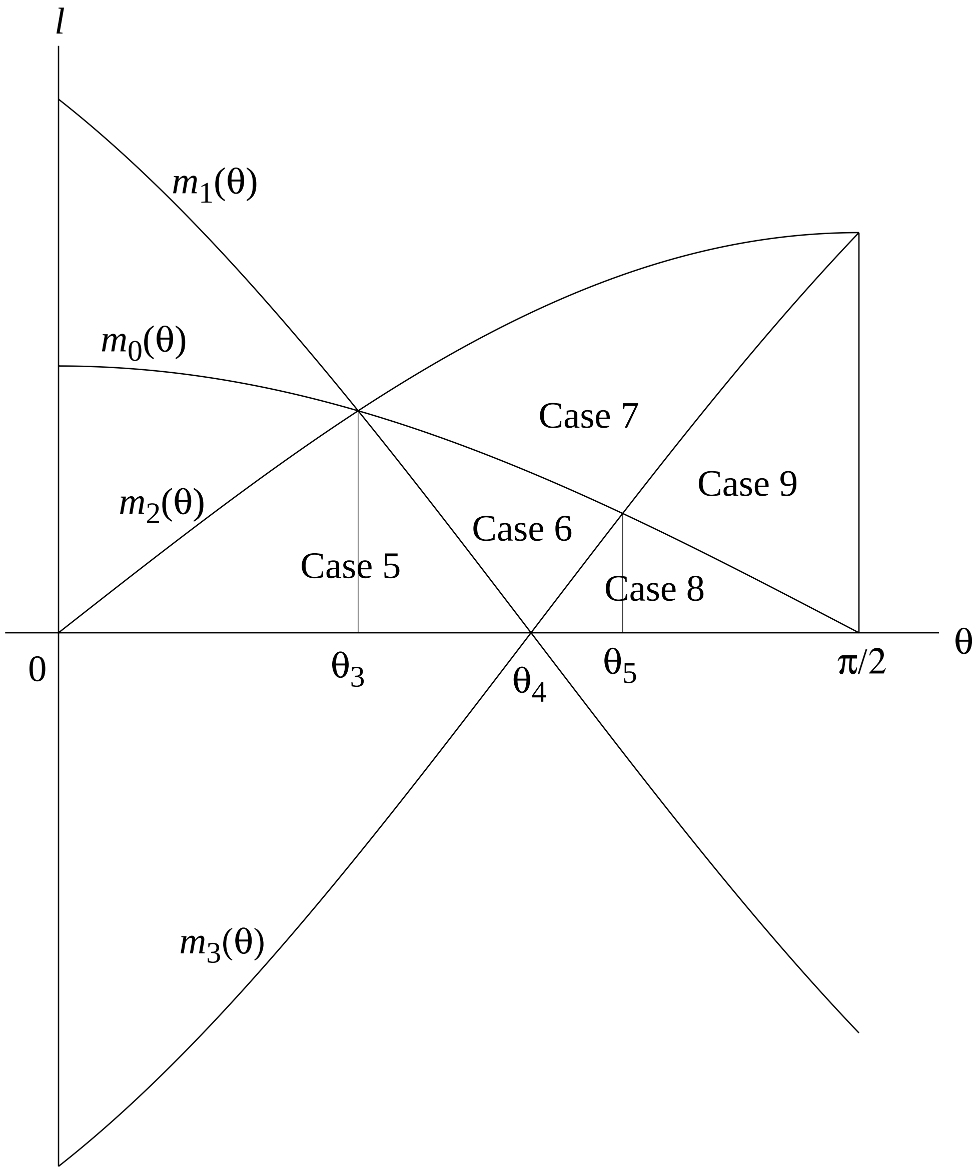

Fig. 10: The (θ, l) ranges for cases 5 through 9.

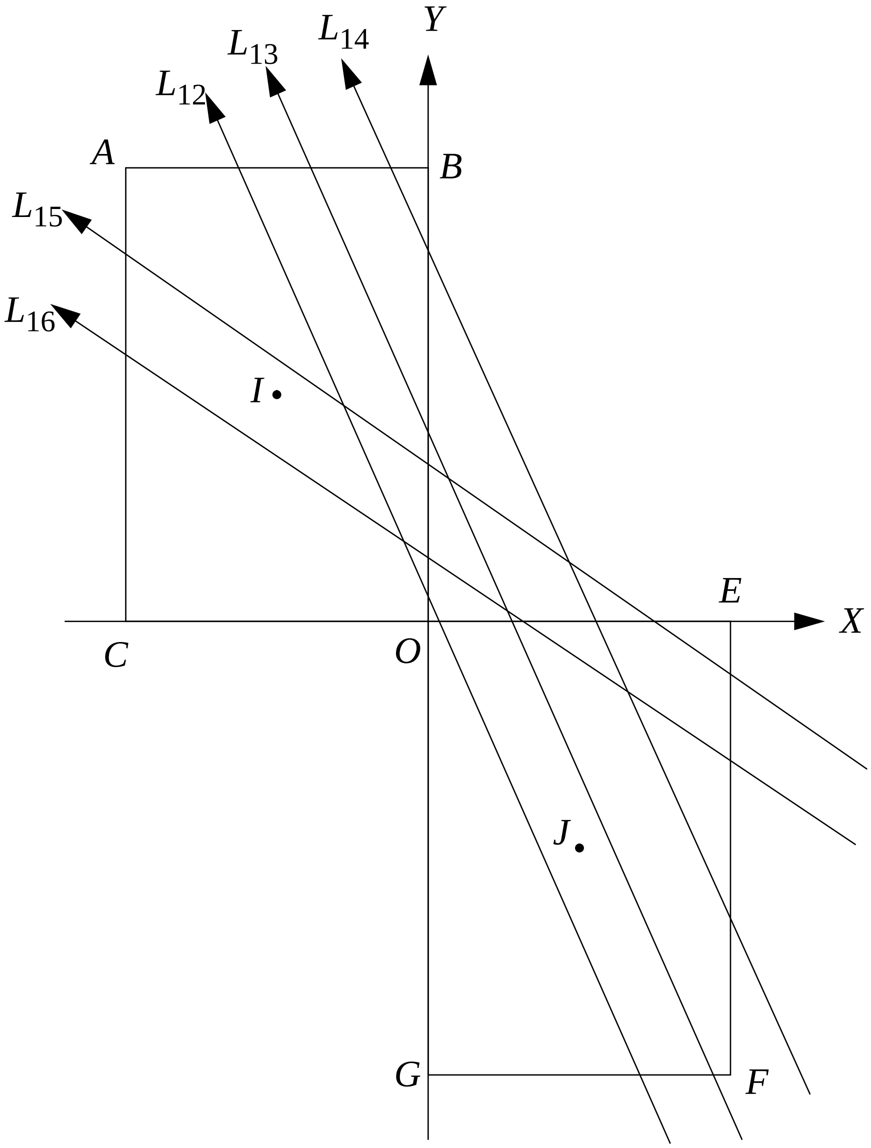

Fig. 11: Lines in cases 12 through 16.

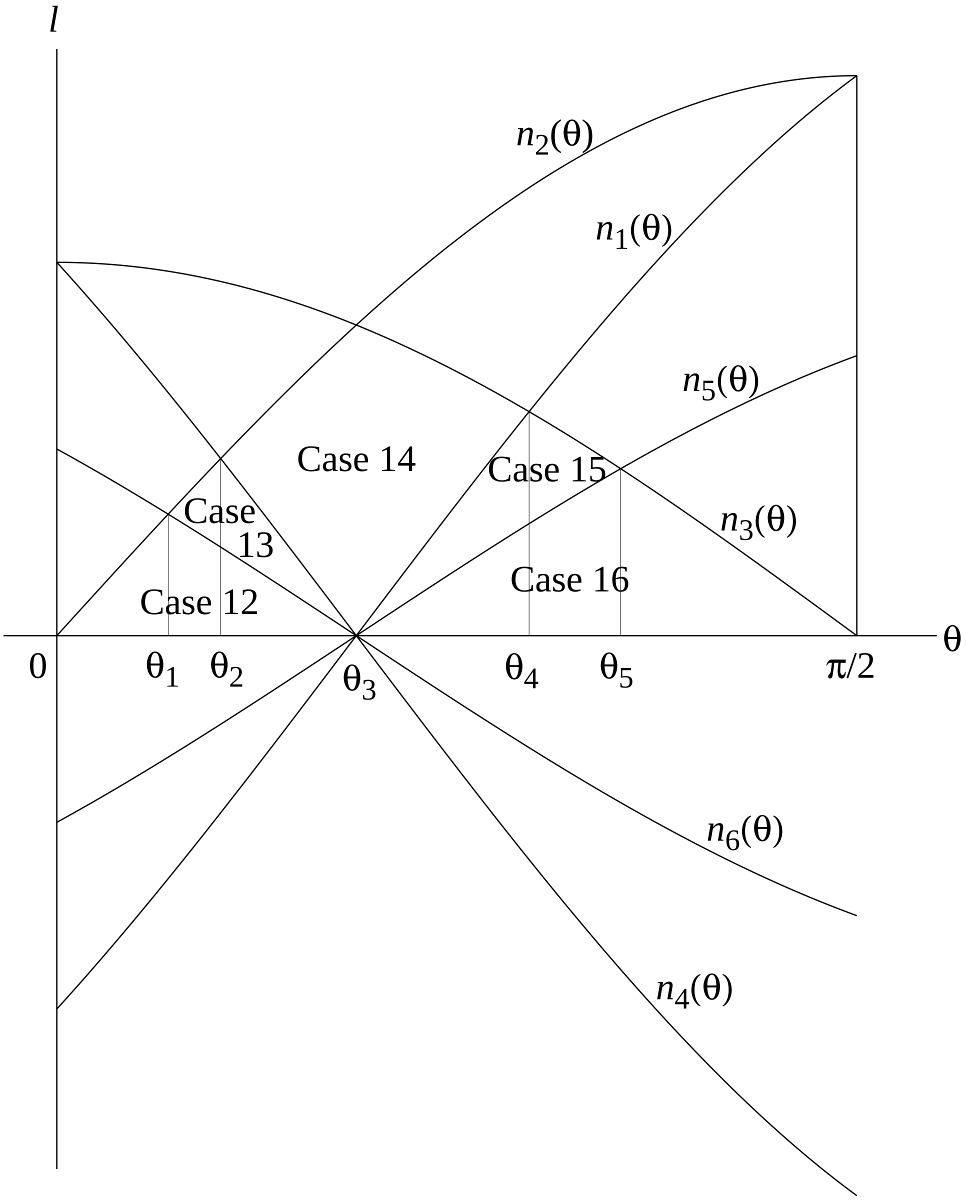

Fig. 12: The (θ, l) ranges for cases 12 through 16.

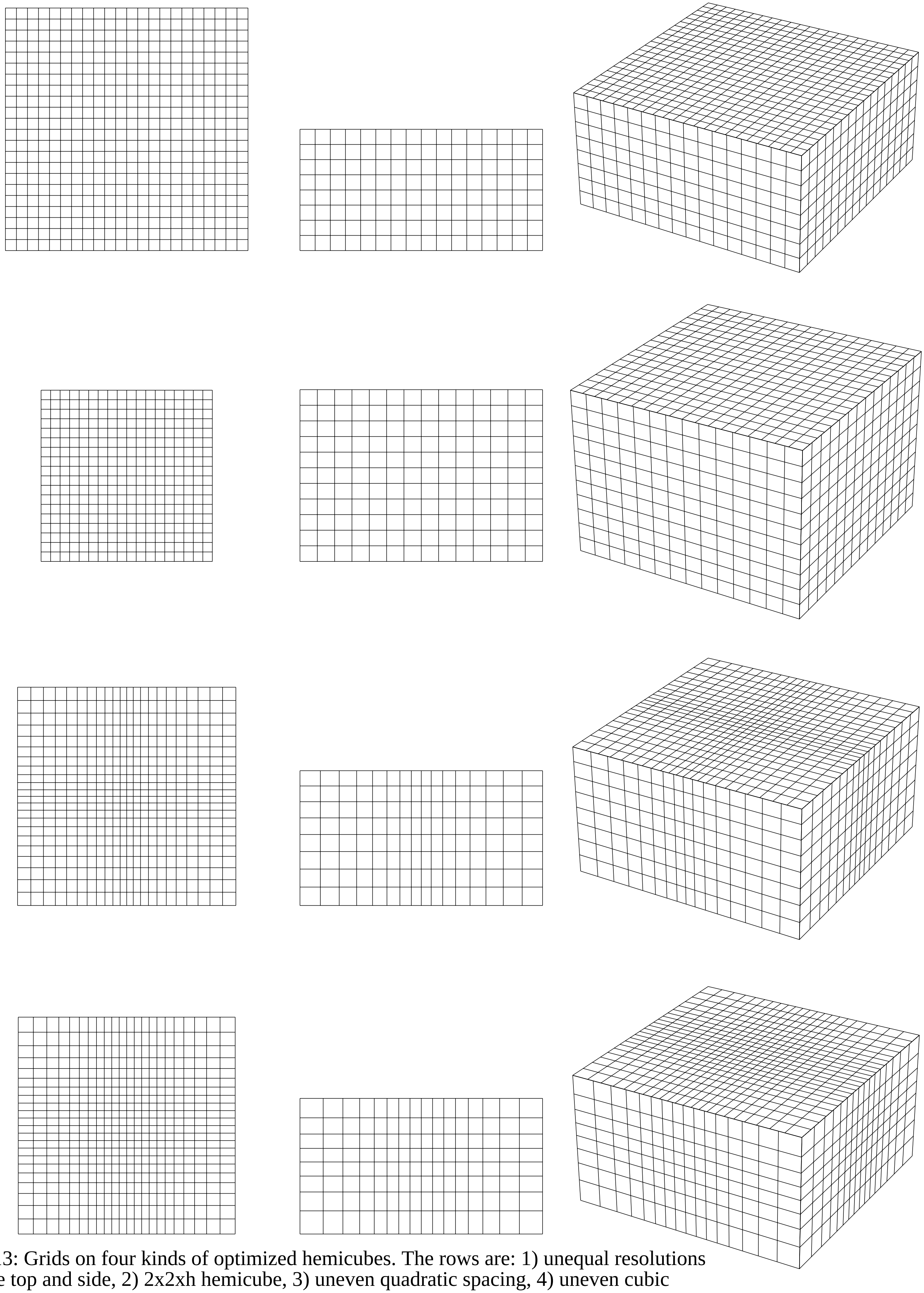

Fig. 13: Grids on four kinds of optimized hemicubes. The rows are: 1) unequal resolutions on the top and side, 2) 2x2xh hemicube, 3) uneven quadratic spacing, 4) uneven cubic spacing.

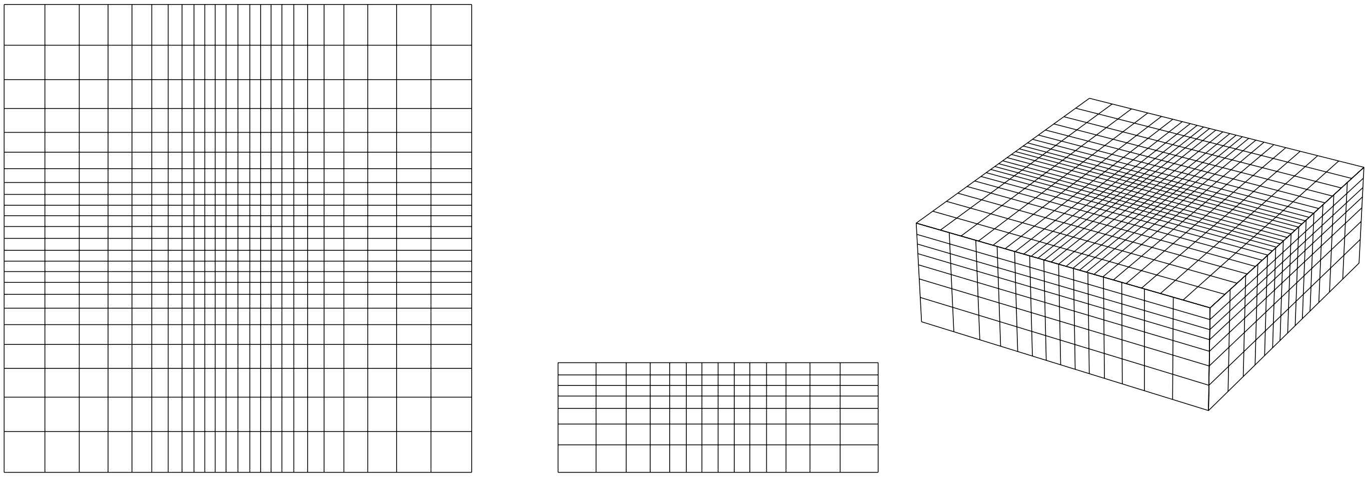

Fig. 14: Grid on hemicube making pixel form factors as equal as possible.